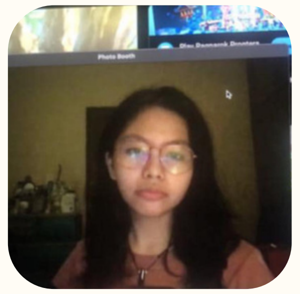
Bite & Byte

Members:

Zergio Achillis Hablado

Jerry Matthew Larracas


Ramelissa Melith Limpio

Princess Iana Mejia

Chloe Nickheala Orzal

Introduction

The ordering process is a difficulty for many fast food restaurants. Due to the manual ordering process, which may be exhausting and perplexing for both staff and consumers, customers frequently have to wait in line for a very long time, make mistakes with their orders, and experience delays when there are too many clients at once.

The project focuses on developing a fast food ordering system that enables customers to place orders more quickly and efficiently in order to address these problems. The system has separate access for the manager and cashier to better manage jobs, a stocking system to keep track of available items, and a menu display. To make it more convenient, the system will also display the approximate wait time for each customer's order and permit the use of coupons or discounts.

Objectives

The goal of this project is to develop a C++ based fast food ordering system that automates the ordering process to minimize manual errors. Develop a C++ program that can:

1. Display a menu and ordering interface that allows the user to order Burger Steak, Coca-Cola and Chicken Nuggets
2. Store and display an order summary showing item details, item quantity and total bill of the customer.
3. Implement a stocking system that tracks product availability and updates after each transaction.

Objectives

4. Create an employee access system, where the cashier handles orders, and the manager can change the available stocks and turn off the system.

5. Integrate a coupon and discount functionalities that apply special discounts such as 20% for PWD or senior citizen and 10% for coupons to the total bill.
6. Calculate and display an estimated waiting time based on the number of items ordered and the ongoing orders before.

Flowchart of the System

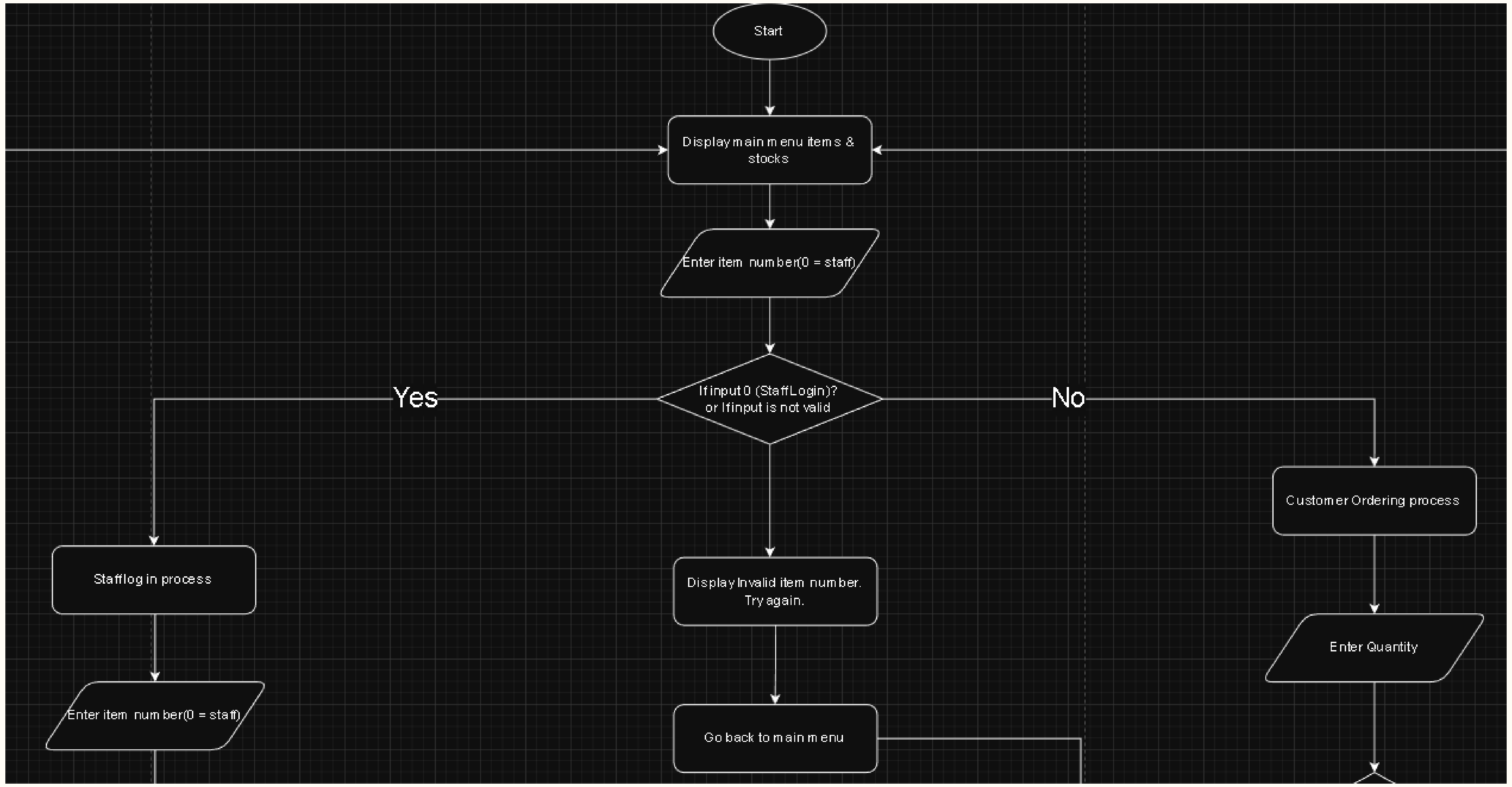

Start
Display main menu items & stocks
Enter item number(0 = staff)
If input 0 (StaffLogin)? or If input is not valid
Yes
No
Staff log in process
Enter item number(0 = staff)
Display Invalid item number. Try again.
Go back to main menu
Customer Ordering process
Enter Quantity

Figure 1: Flow chart of Fast Food Ordering System

Flowchart of the System

Figure 1 illustrates the start of the system. The system will display main menu items and available stocks. The system then asks the user to enter an item number. If the input is invalid, then the system will go back to the main menu and if the input is 0, the system will go to the staff log in process. But if the user inputs a valid number, then it will undergo a customer ordering process.

Flowchart of the System

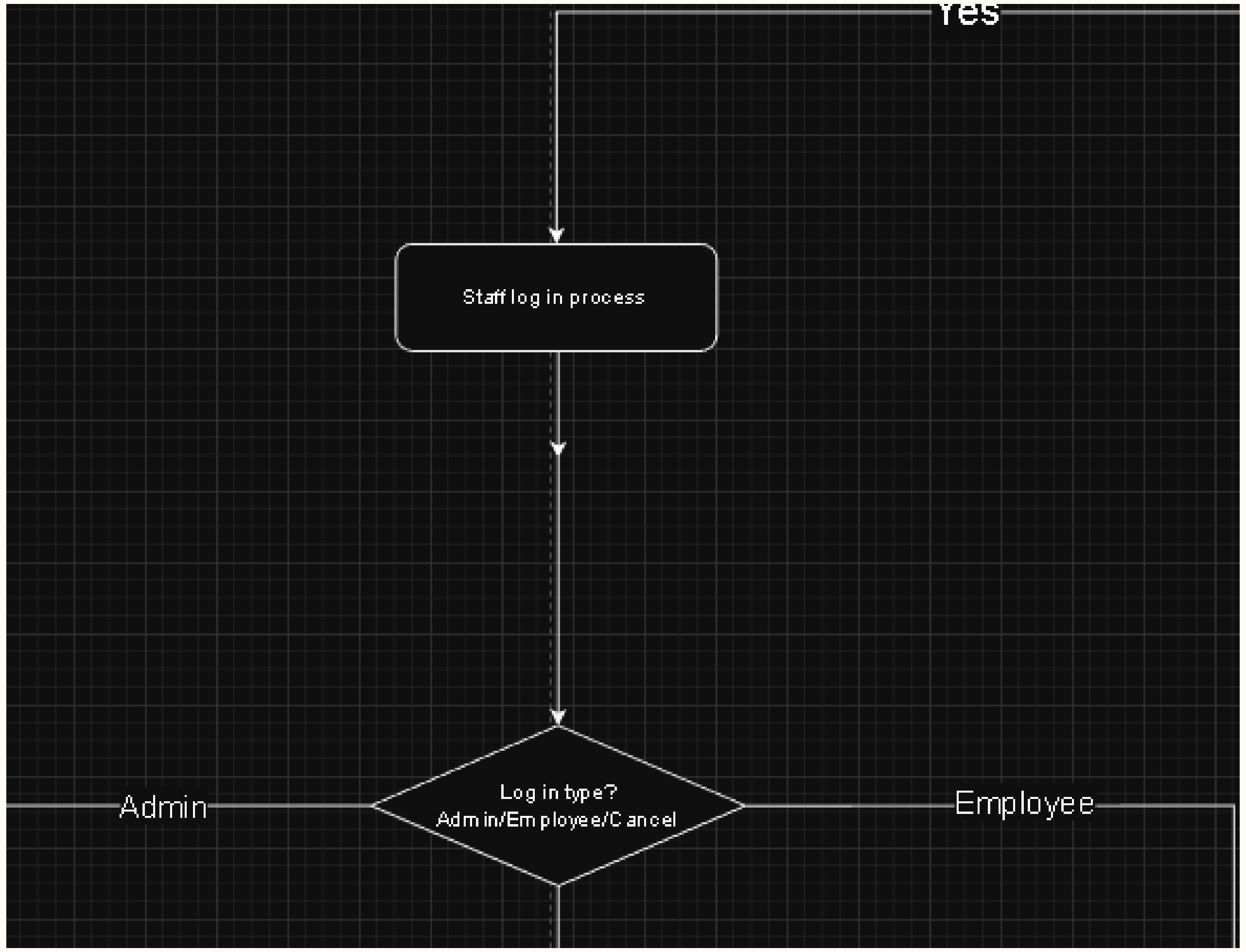


Figure 2: Flowchart part of Staff log in process

Flowchart of the System

Figure 2 shows that if the input is 0, then the system will proceed to the staff log in process. Then the system will then ask the user what type of login if it is admin or employee or cancel the log in process.

Flowchart of the System

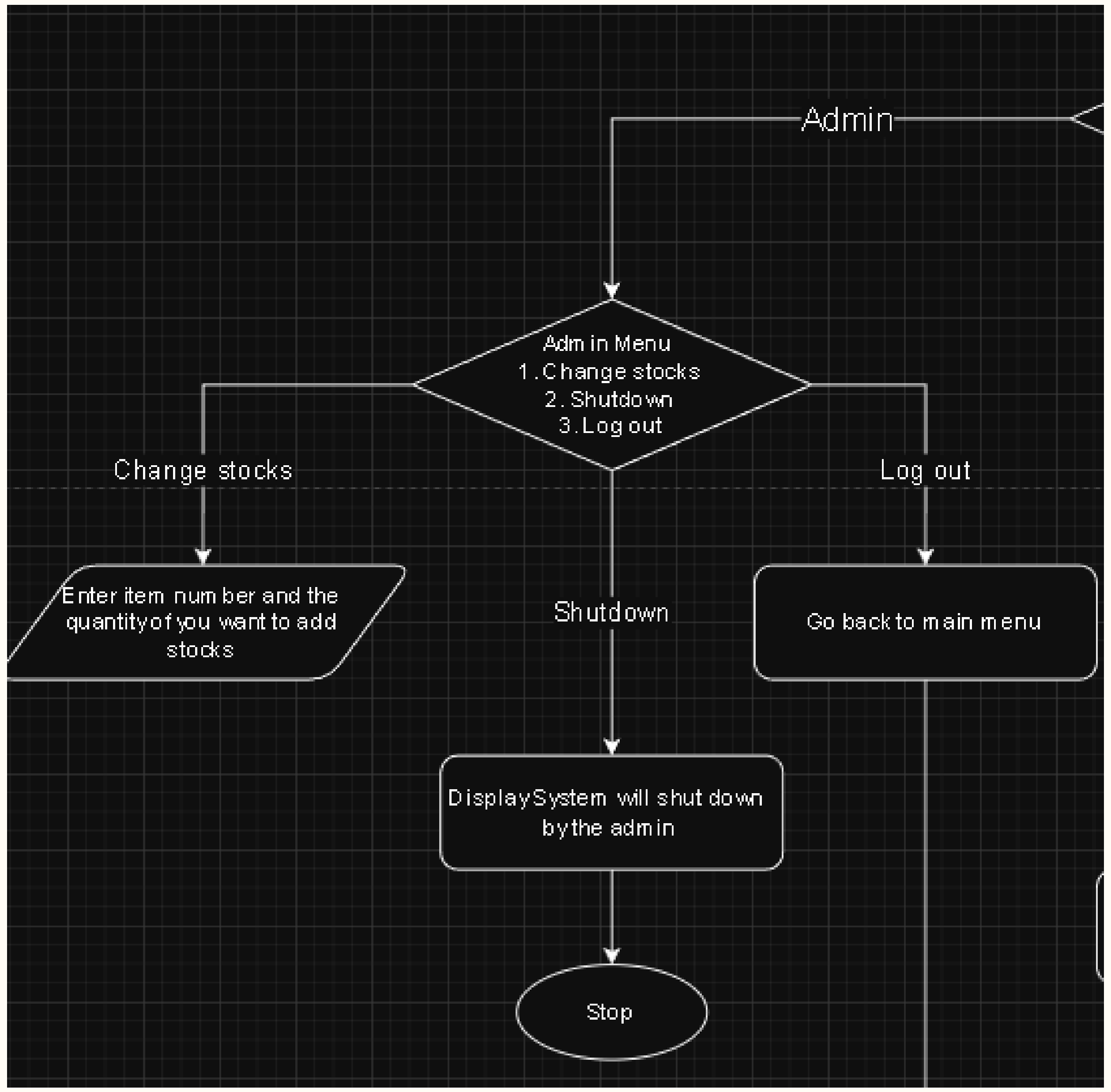


Figure 3: Flow chart part of admin log in process and its functions

Flowchart of the System

Figure 3 shows that if the user is logged in as an admin, the system will display an admin menu and has a function to change stocks, shutdown and log out as an admin

Flowchart of the System

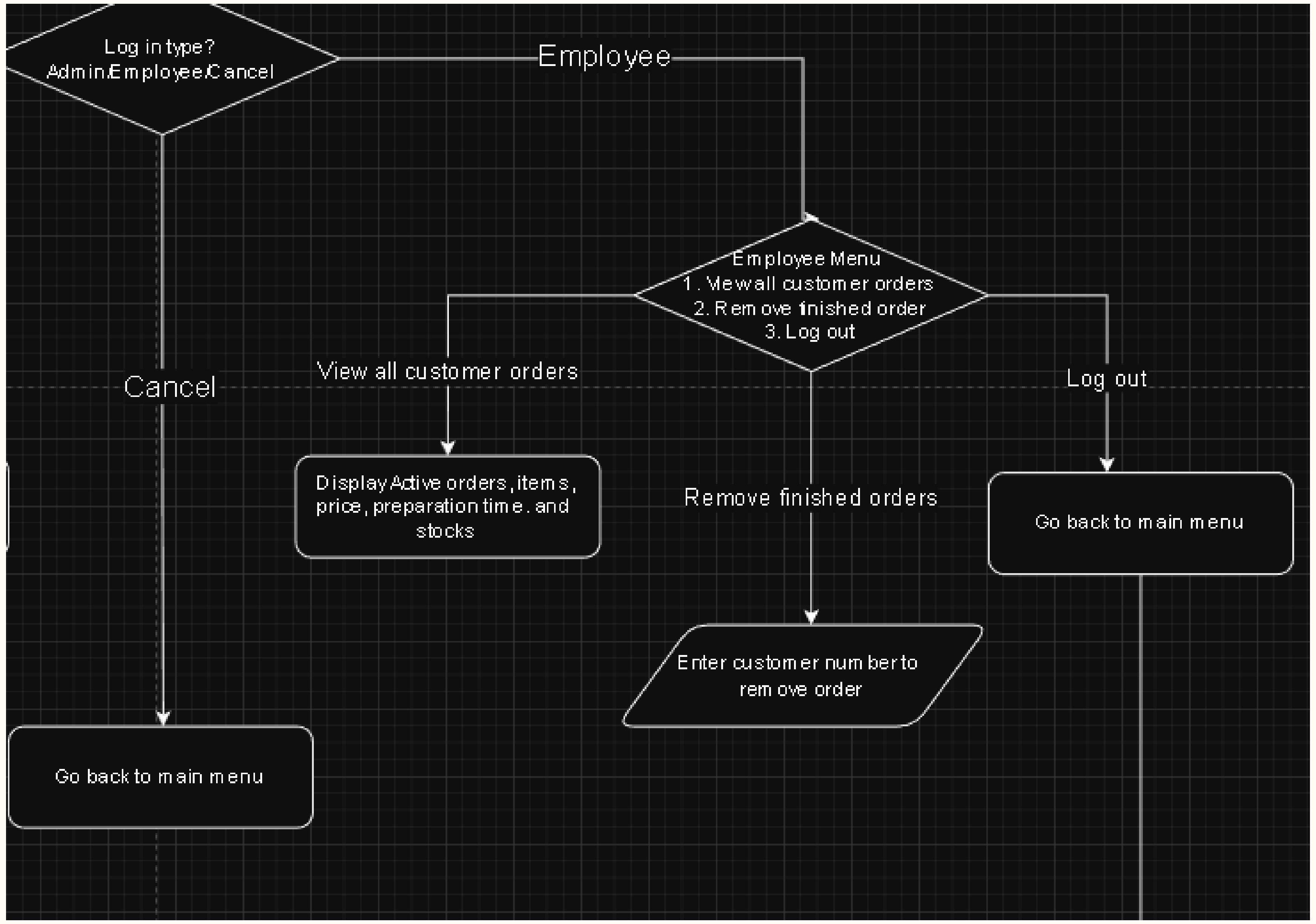

Log in type? Admin/Employee/Cancel
Employee
Employee Menu 1. View all customer orders 2. Remove finished order 3. Log out
View all customer orders
Cancel
Log out
Display Active orders, items, price, preparation time. and stocks
Remove finished orders
Go back to main menu
Enter customer number to remove order
Go back to main menu

Figure 4: Flow chart part of employee log in process and its functions.

Flowchart of the System

Figure 4 represents that if the user is logged in as an employee, the system will display an employee menu that has a function to view all customer orders, remove finished orders, and log out as an employee.

Flowchart of the System

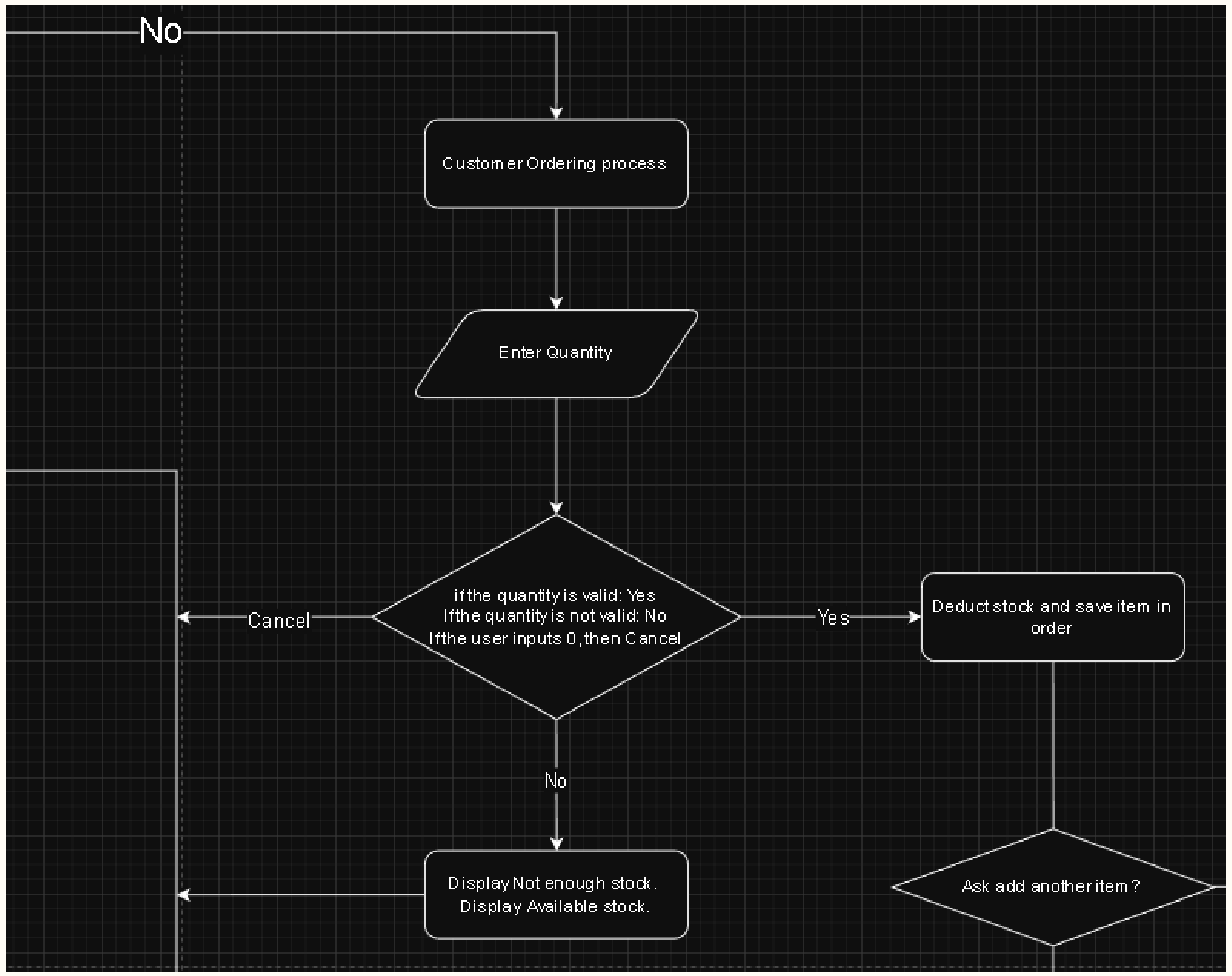

No
Customer Ordering process
Enter Quantity
if the quantity is valid: Yes
If the quantity is not valid: No
If the user inputs 0, then Cancel
Cancel
Yes
Deduct stock and save item in order
No
Display Not enough stock.
Display Available stock.
Ask add another item?

Figure 5: Flow chart part of customer ordering process

Flowchart of the System

Figure 5 shows if the input is valid, then the system will proceed to the customer ordering process. The system will then ask the user to enter a quantity. If the quantity is valid, the system will deduct the stock of the selected item and save the item in the order list. If the quantity is not valid, the system will display not enough stock and shows the only available stock. And if the user inputs 0, the system will return to the main menu.

Flowchart of the System

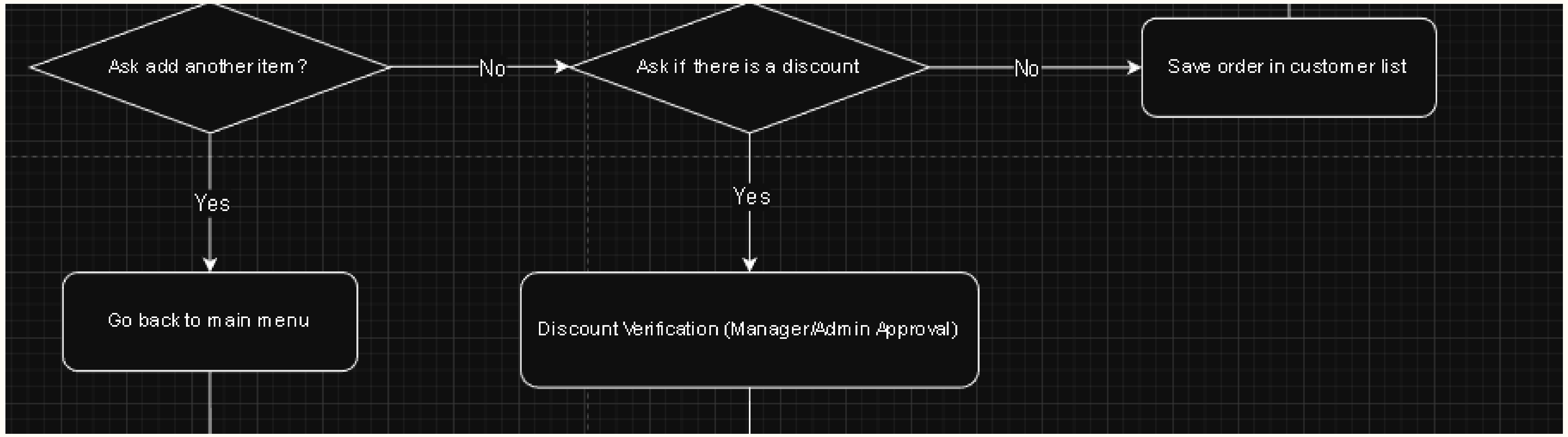

Ask add another item?
No
Ask if there is a discount
No
Save order in customer list
Yes
Yes
Go back to main menu
Discount Verification (Manager/Admin Approval)

Figure 6: Flow chart part of customer ordering process and discount verification.

Flowchart of the System

Figure 6 shows that if the item successfully deducted and saved in the order list, the system will then ask the user to add another item or not. If yes, then the system will return to the main menu and if no, the system will then ask again if there is a discount. If not, the system will save the order in the customer list with the complete receipt, including prices and estimated time. And if yes, the system will then proceed to the discount verification process.

Flowchart of the System

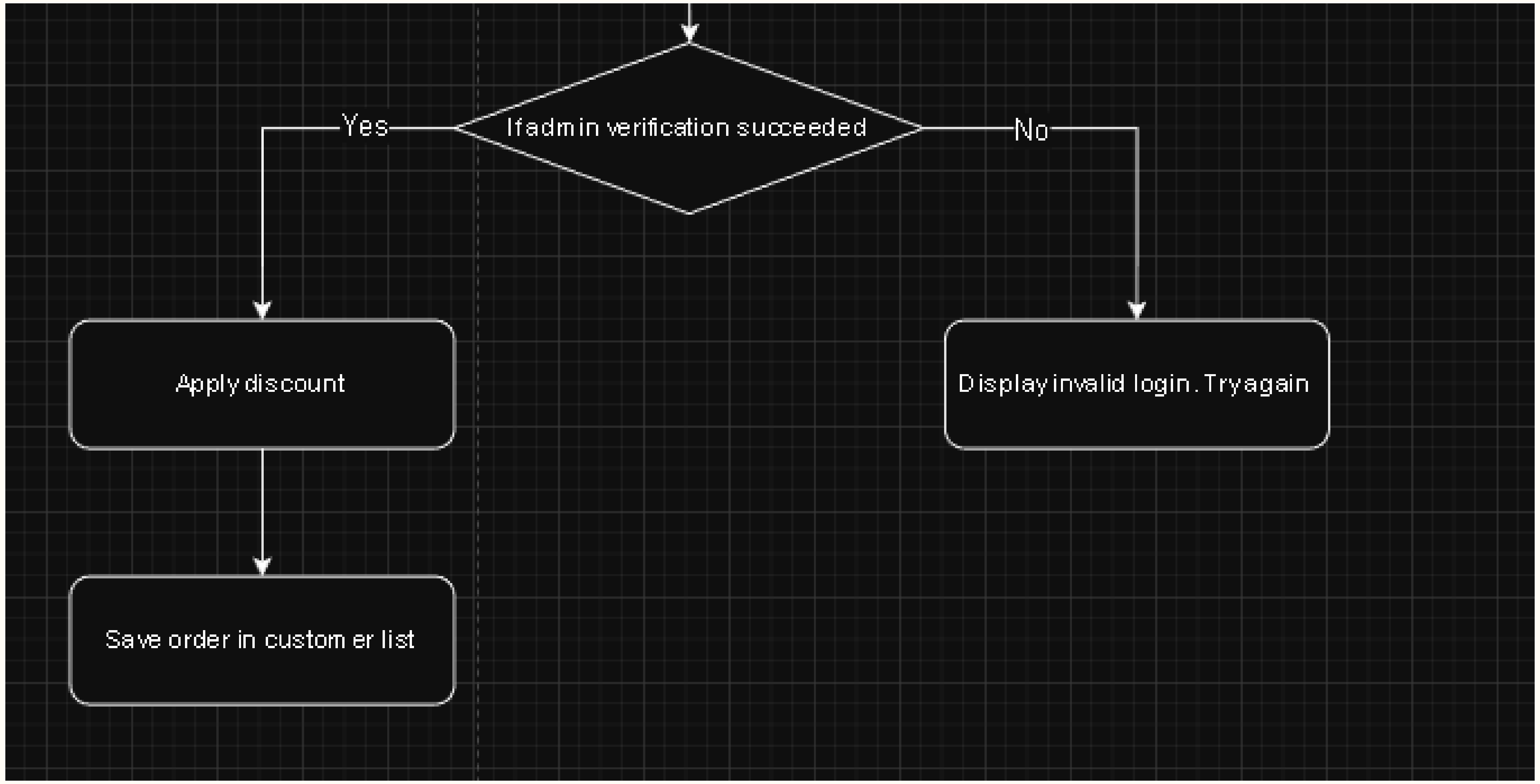

If admin verification succeeded
Yes
No
Apply discount
Display invalid login. Try again
Save order in customer list

Figure 7: Flow chart part of discount verification process

Flowchart of the System

Figure 7 shows the discount verification process where the admin or manager only has access to approve the discount.

Pseudocode

```
START

Create menu items, prices, preparation time, and stocks
Create empty list of customer orders
Set customer count to 0
Set system status to running

REPEAT while system is running:
    Create a new order for a customer
    Set discount to 0% (No discount yet)
```

Pseudocode

```
DO the following while ordering:

    DISPLAY menu items, prices, and remaining stocks
            Ask user to select an item
            Ask user for quantity
                IF quantity is 0:
                    Cancel and return to menu
                IF quantity is more than stocks:
                    Display error and ask again
                END IF

    Deduct stock from item
    Add item, quantity, price, and preparation time to customer
order

    Ask if the customer want to add another item
        IF yes → repeat ordering
        IF no → proceed
        END IF
```

Pseudocode

```
    IF user enters 0:
        Go to staff login

    IF admin shuts down system
        stop process
   OTHERWISE → go back to menu
   END IF

    IF item is not valid:
        Display error and ask again
    END IF

END DO

IF system was shut down → STOP process
```

Pseudocode

```
Ask if the customer has discount
    IF yes:
        Ask discount type (Senior/PWD = 20%, Coupon = 10%)
        Ask manager/admin to login for approval
            IF login success:
                Apply discount
            IF login failed:
                Do not apply discount
    END IF
```

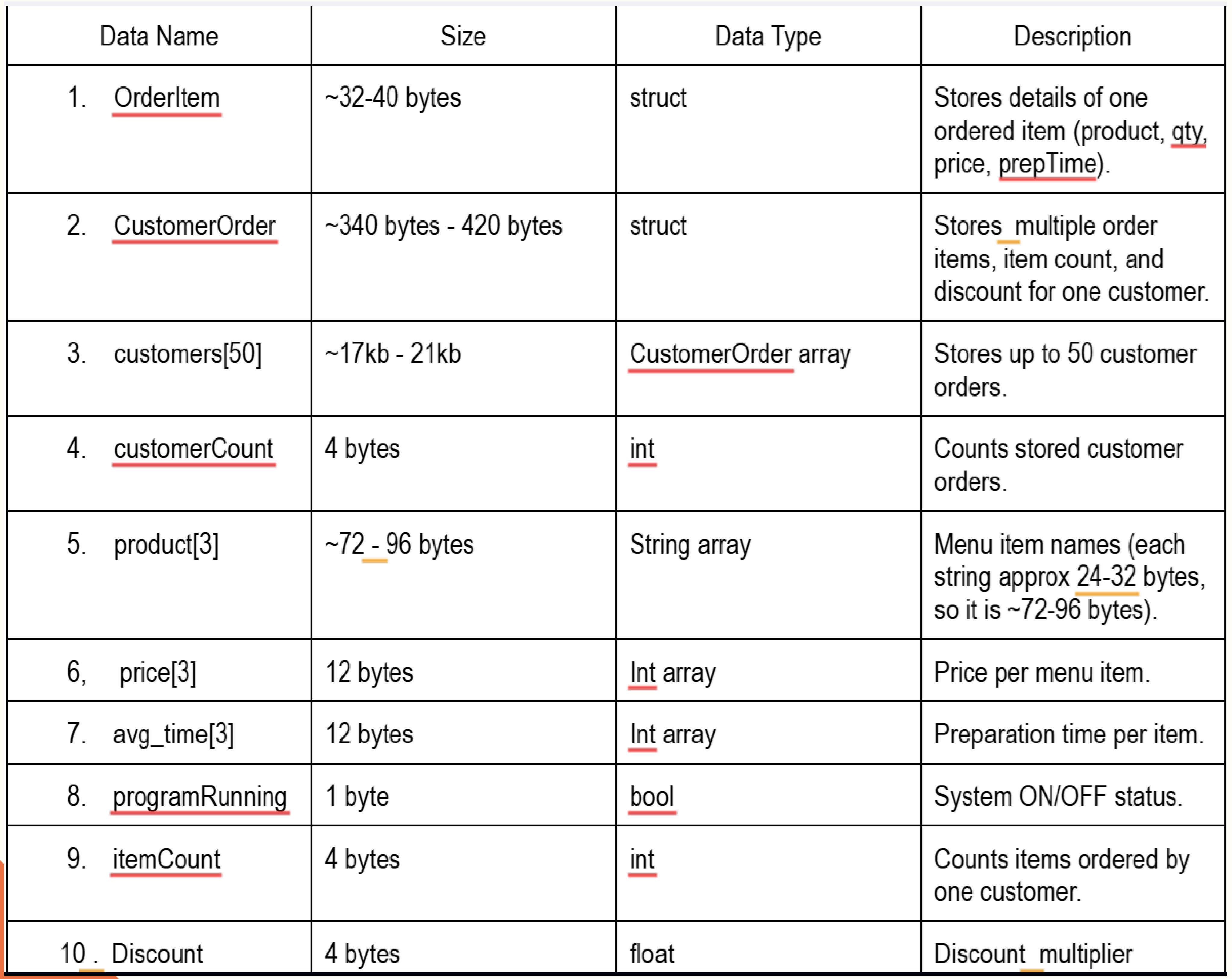
Pseudocode

```
Save order to customer list
Increase customer count
Display all active orders

END REPEAT


WHEN admin shuts down:
    DISPLAY  “System stopped by admin”


END
```

Pseudocode

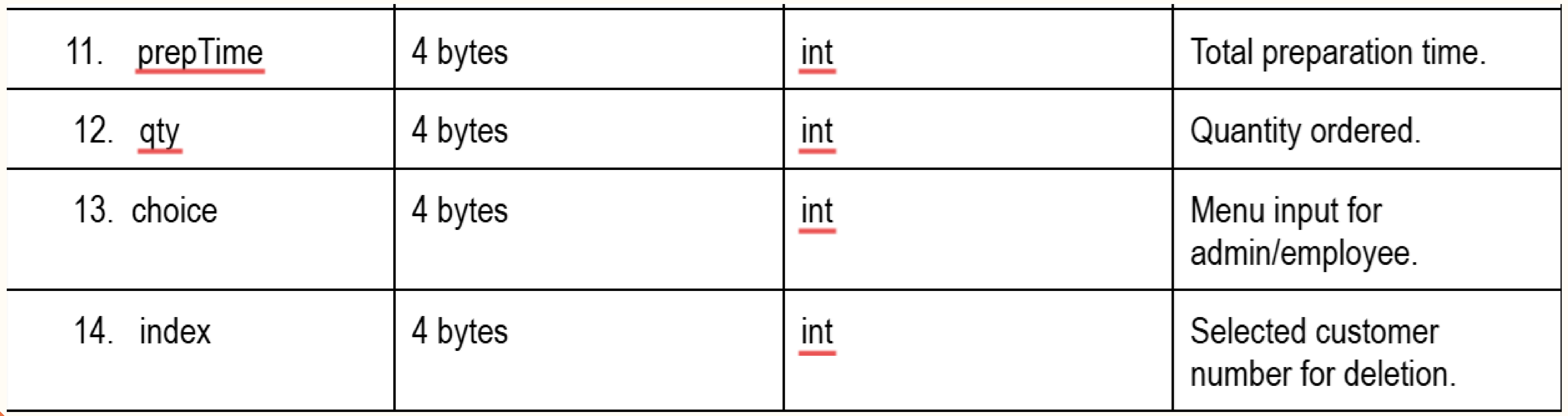
```
Ask if login is Admin or Employee
IF Admin:
    Ask username & password
IF correct → open Admin menu
    ELSE Go back to the main menu and display error
END IF

IF Employee:
    Ask username & password
    IF correct → open Employee menu
    ELSE Go back to the main menu and display error
END IF
```

Data Dictionary



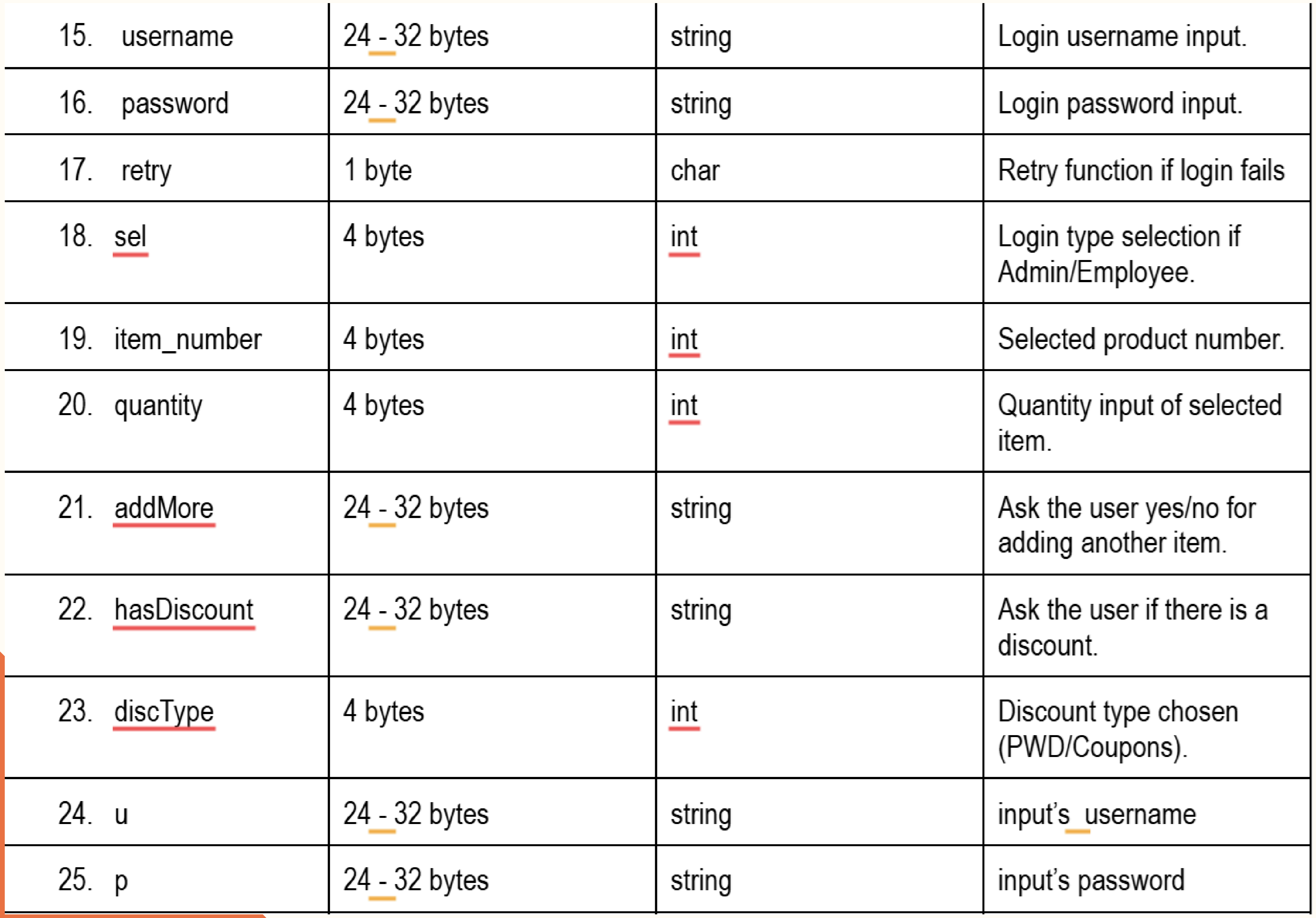
Data Name	Size	Data Type	Description
1. OrderItem	~32-40 bytes	struct	Stores details of one ordered item (product, qty, price, prepTime).
2. CustomerOrder	~340 bytes - 420 bytes	struct	Stores multiple order items, item count, and discount for one customer.
3. customers[50]	~17kb - 21kb	CustomerOrder array	Stores up to 50 customer orders.
4. customerCount	4 bytes	int	Counts stored customer orders.
5. product[3]	~72 - 96 bytes	String array	Menu item names (each string approx 24-32 bytes, so it is ~72-96 bytes).
6, price[3]	12 bytes	Int array	Price per menu item.
7. avg_time[3]	12 bytes	Int array	Preparation time per item.
8. programRunning	1 byte	bool	System ON/OFF status.
9. itemCount	4 bytes	int	Counts items ordered by one customer.
10 . Discount	4 bytes	float	Discount multiplier

Data Dictionary



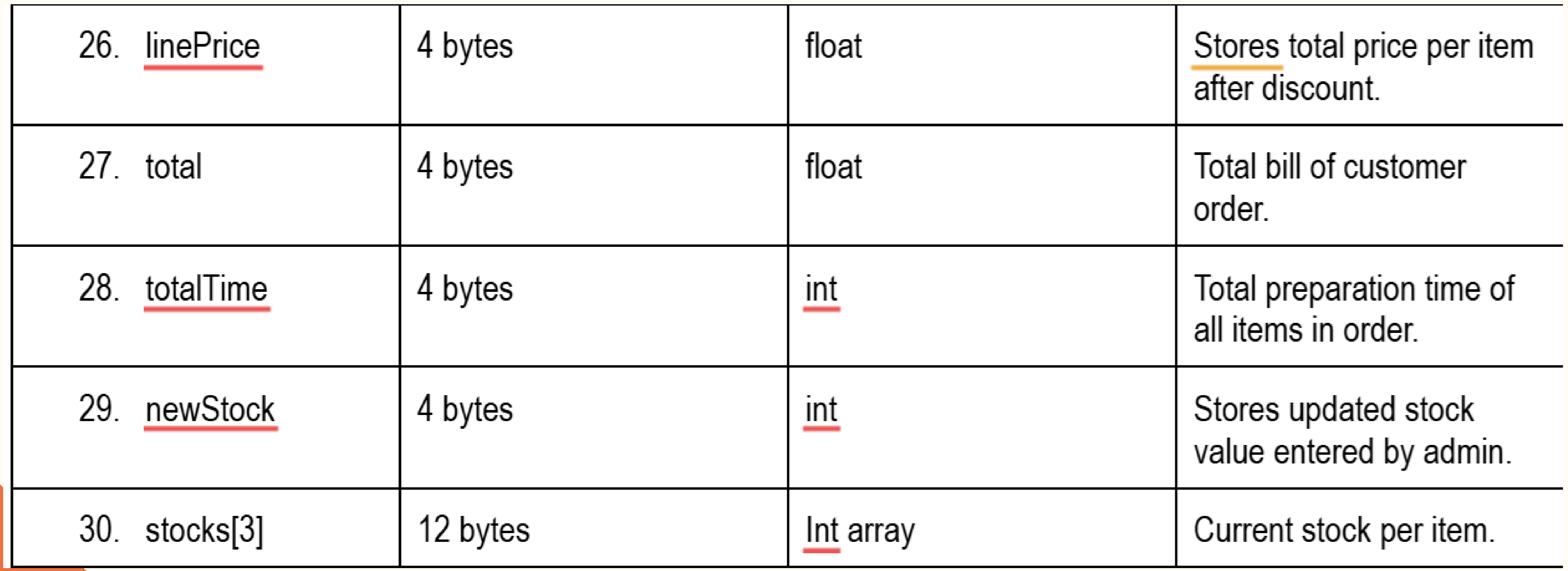
11. prepTime	4 bytes	int	Total preparation time.
12. qty	4 bytes	int	Quantity ordered.
13. choice	4 bytes	int	Menu input for admin/employee.
14. index	4 bytes	int	Selected customer number for deletion.

Data Dictionary



15. username	24 - 32 bytes	string	Login username input.
16. password	24 - 32 bytes	string	Login password input.
17. retry	1 byte	char	Retry function if login fails
18. sel	4 bytes	int	Login type selection if Admin/Employee.
19. item_number	4 bytes	int	Selected product number.
20. quantity	4 bytes	int	Quantity input of selected item.
21. addMore	24 - 32 bytes	string	Ask the user yes/no for adding another item.
22. hasDiscount	24 - 32 bytes	string	Ask the user if there is a discount.
23. discType	4 bytes	int	Discount type chosen (PWD/Coupons).
24. u	24 - 32 bytes	string	input's username
25. p	24 - 32 bytes	string	input's password

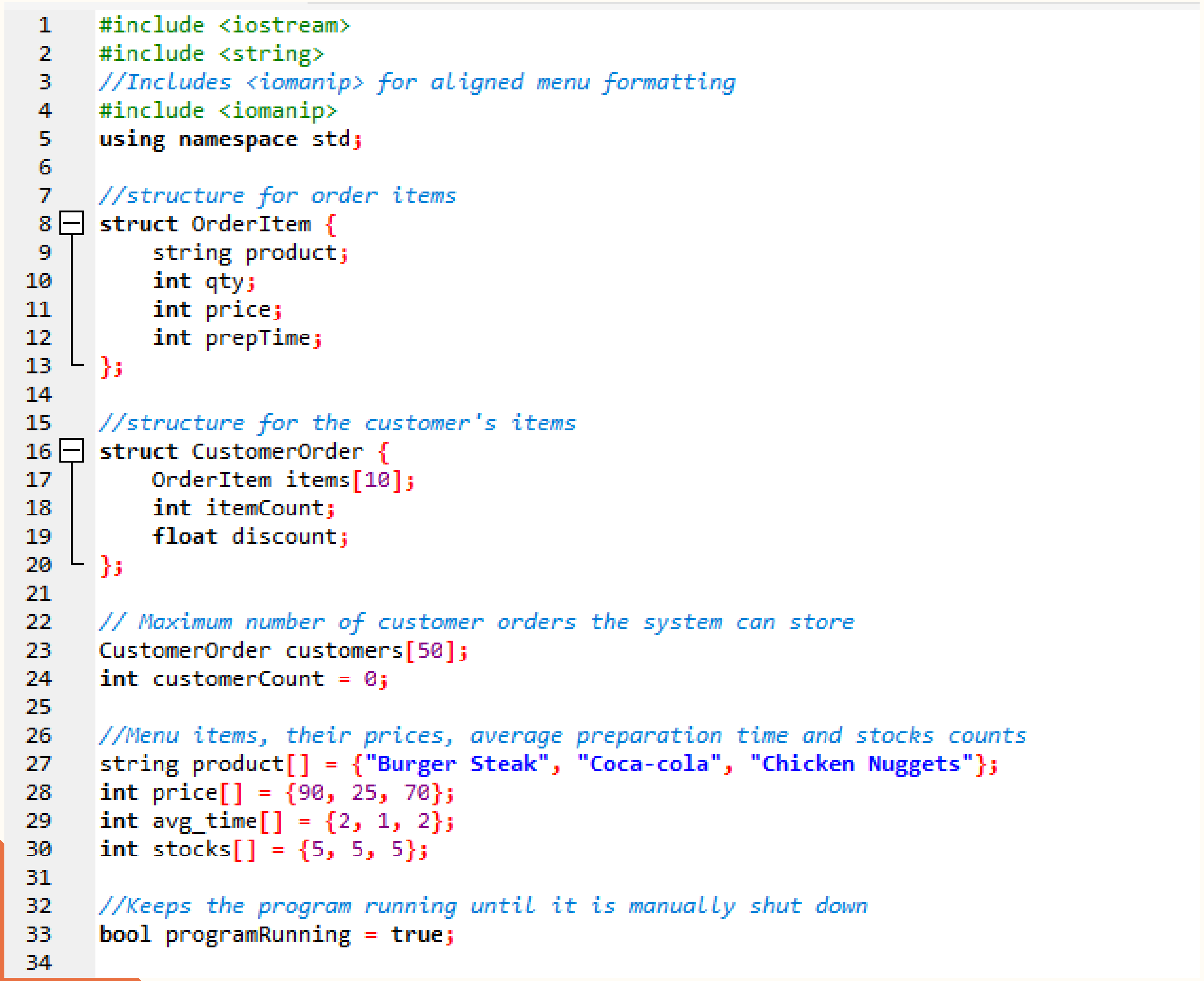
Data Dictionary



26. linePrice	4 bytes	float	Stores total price per item after discount.
27. total	4 bytes	float	Total bill of customer order.
28. totalTime	4 bytes	int	Total preparation time of all items in order.
29. newStock	4 bytes	int	Stores updated stock value entered by admin.
30. stocks[3]	12 bytes	Int array	Current stock per item.

Code

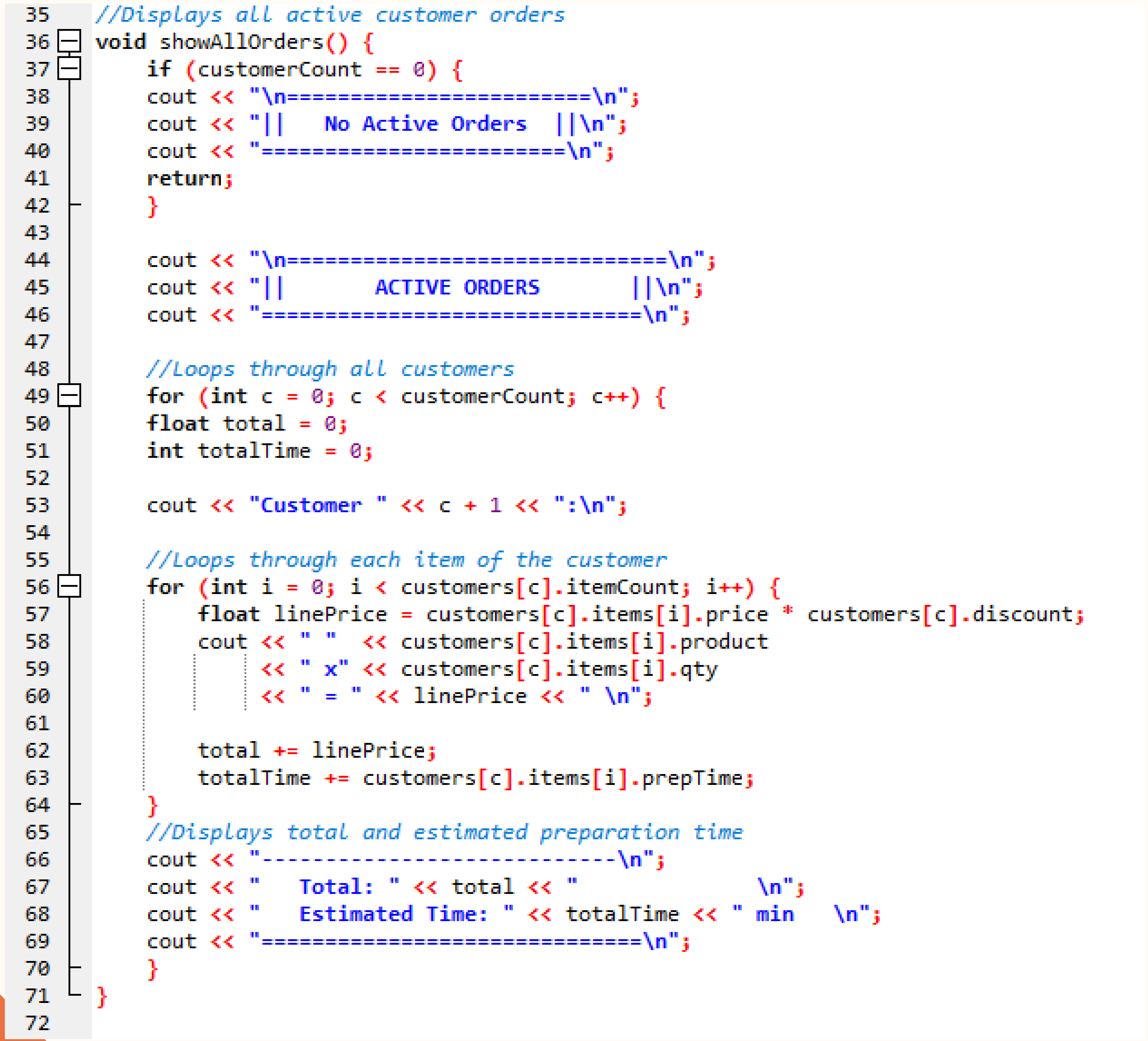
Code



```cpp
#include <iostream>
#include <string>
//Includes <iomanip> for aligned menu formatting
#include <iomanip>
using namespace std;

//structure for order items
struct OrderItem {
    string product;
    int qty;
    int price;
    int prepTime;
};

//structure for the customer's items
struct CustomerOrder {
    OrderItem items[10];
    int itemCount;
    float discount;
};

// Maximum number of customer orders the system can store
CustomerOrder customers[50];
int customerCount = 0;

//Menu items, their prices, average preparation time and stocks counts
string product[] = {"Burger Steak", "Coca-cola", "Chicken Nuggets"};
int price[] = {90, 25, 70};
int avg_time[] = {2, 1, 2};
int stocks[] = {5, 5, 5};

//Keeps the program running until it is manually shut down
bool programRunning = true;

```

Figure 10. Initial Setups

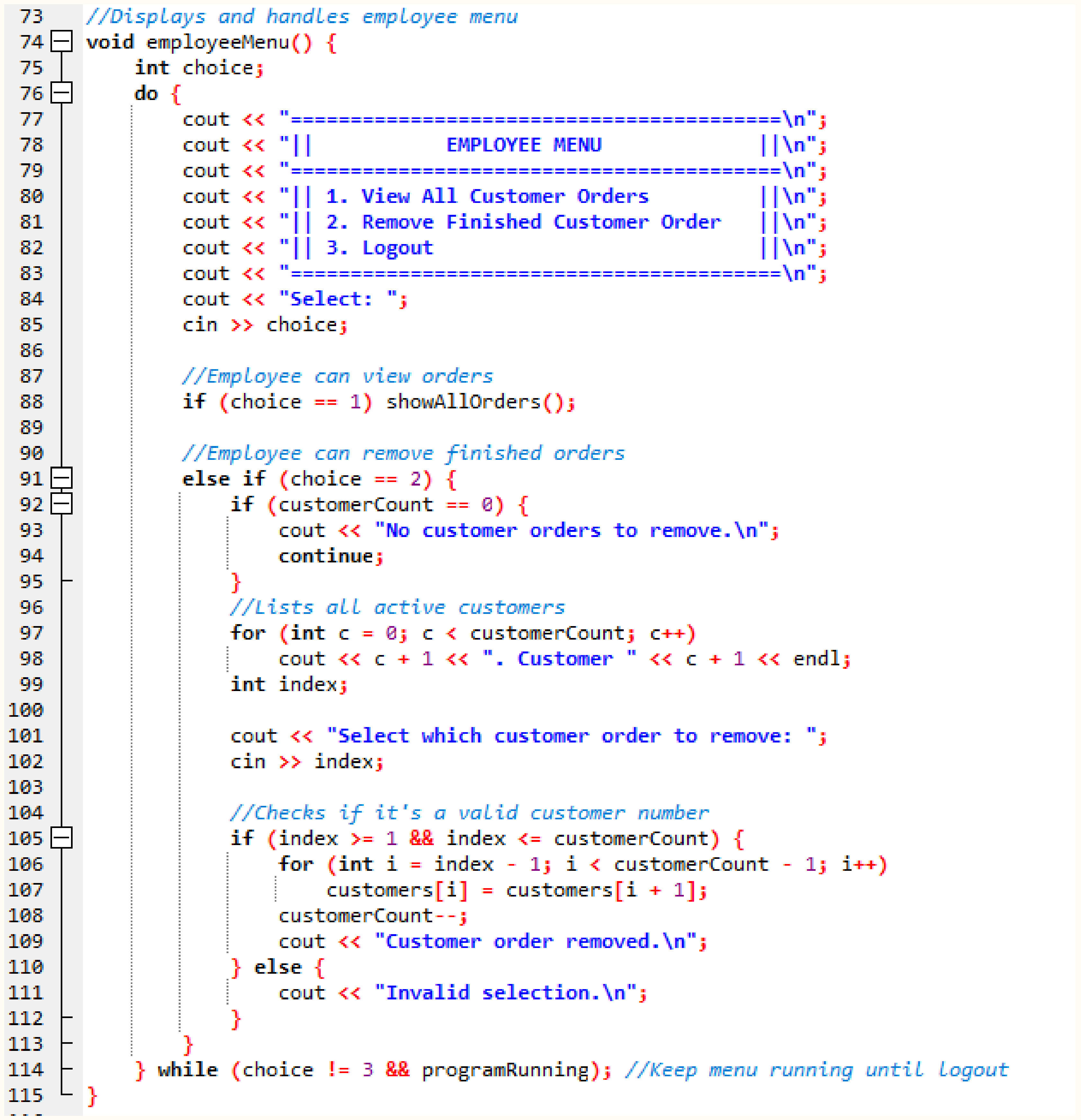
CODE



```
//Displays all active customer orders
void showAllOrders() {
    if (customerCount == 0) {
    cout << "\n========================\n";
    cout << "||   No Active Orders   ||\n";
    cout << "========================\n";
    return;
    }

    cout << "\n==============================\n";
    cout << "||        ACTIVE ORDERS       ||\n";
    cout << "==============================\n";

    //Loops through all customers
    for (int c = 0; c < customerCount; c++) {
    float total = 0;
    int totalTime = 0;

    cout << "Customer " << c + 1 << ":\n";

    //Loops through each item of the customer
    for (int i = 0; i < customers[c].itemCount; i++) {
        float linePrice = customers[c].items[i].price * customers[c].discount;
        cout << " "  << customers[c].items[i].product
             << " x" << customers[c].items[i].qty
             << " = " << linePrice << " \n";

        total += linePrice;
        totalTime += customers[c].items[i].prepTime;
    }
    //Displays total and estimated preparation time
    cout << "----------------------------\n";
    cout << "   Total: " << total << "              \n";
    cout << "   Estimated Time: " << totalTime << " min   \n";
    cout << "==============================\n";
    }
}
```

Figure 11. Show all orders

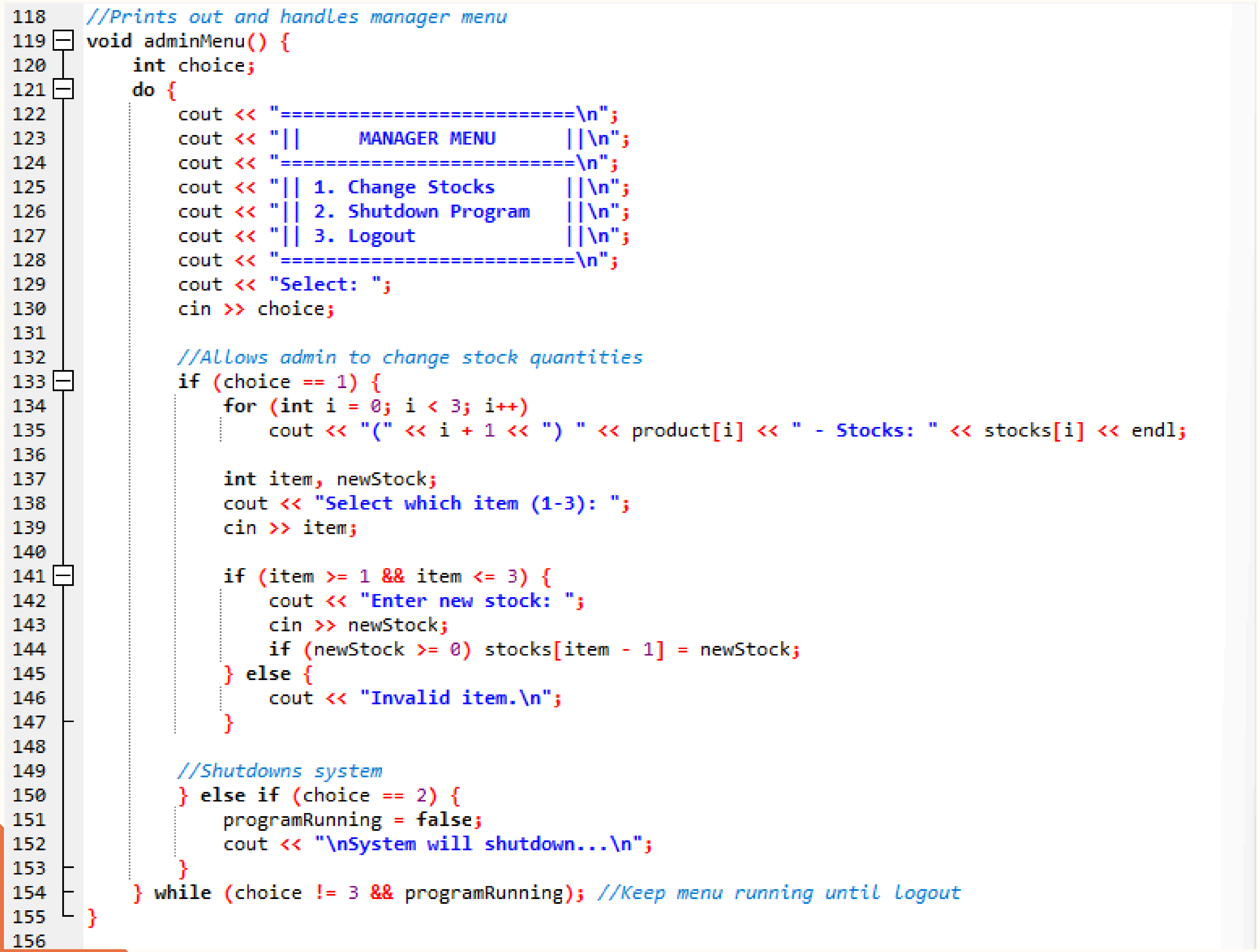
CODE



```cpp
//Displays and handles employee menu
void employeeMenu() {
    int choice;
    do {
        cout << "=========================================\n";
        cout << "||            EMPLOYEE MENU            ||\n";
        cout << "=========================================\n";
        cout << "|| 1. View All Customer Orders         ||\n";
        cout << "|| 2. Remove Finished Customer Order   ||\n";
        cout << "|| 3. Logout                           ||\n";
        cout << "=========================================\n";
        cout << "Select: ";
        cin >> choice;

        //Employee can view orders
        if (choice == 1) showAllOrders();

        //Employee can remove finished orders
        else if (choice == 2) {
            if (customerCount == 0) {
                cout << "No customer orders to remove.\n";
                continue;
            }
            //Lists all active customers
            for (int c = 0; c < customerCount; c++)
                cout << c + 1 << ". Customer " << c + 1 << endl;
            int index;

            cout << "Select which customer order to remove: ";
            cin >> index;

            //Checks if it's a valid customer number
            if (index >= 1 && index <= customerCount) {
                for (int i = index - 1; i < customerCount - 1; i++)
                    customers[i] = customers[i + 1];
                customerCount--;
                cout << "Customer order removed.\n";
            } else {
                cout << "Invalid selection.\n";
            }
        }
    } while (choice != 3 && programRunning); //Keep menu running until logout
}
```

Figure 12. Employee Menu

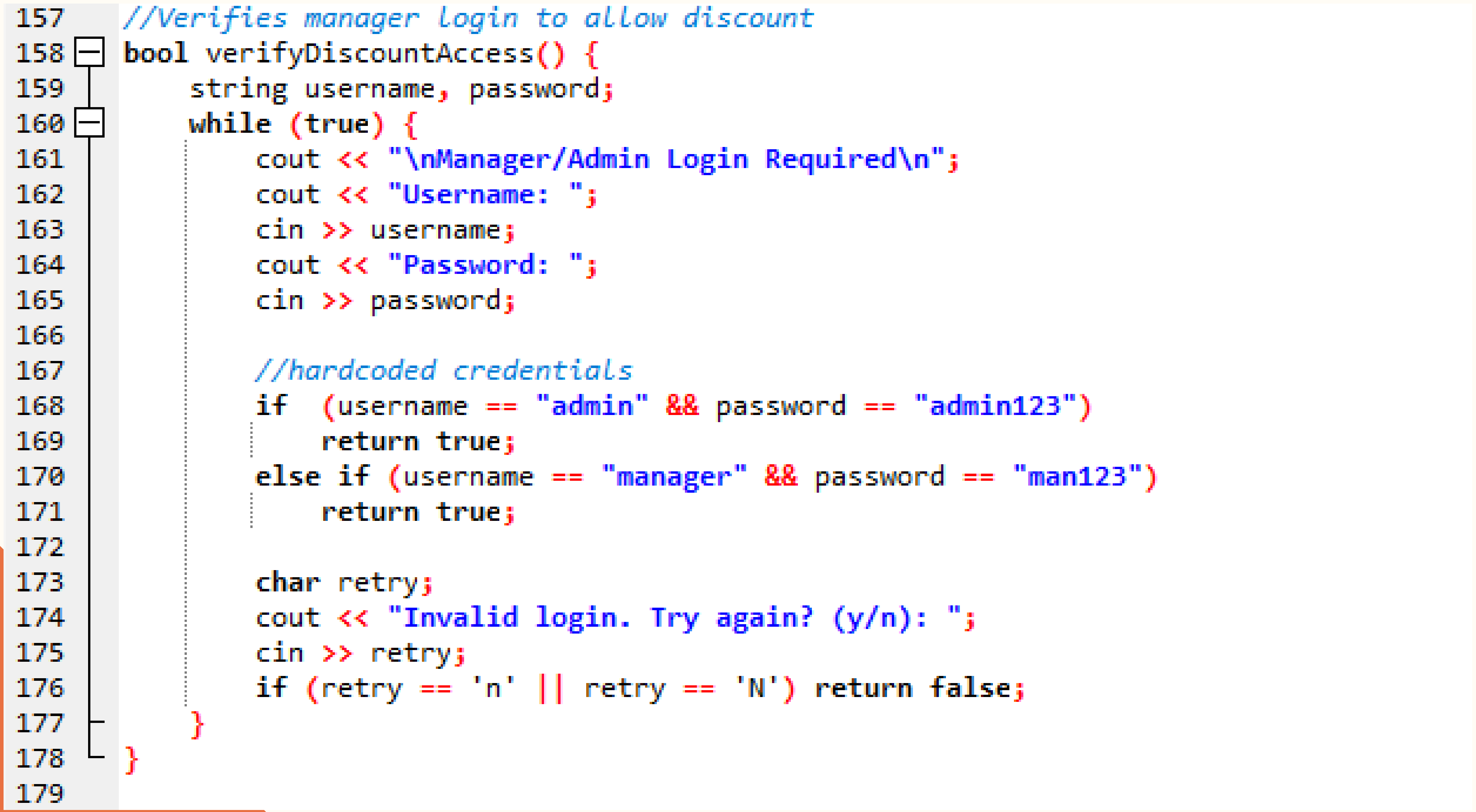
CODE



```cpp
//Prints out and handles manager menu
void adminMenu() {
    int choice;
    do {
        cout << "==========================\n";
        cout << "||     MANAGER MENU      ||\n";
        cout << "==========================\n";
        cout << "|| 1. Change Stocks      ||\n";
        cout << "|| 2. Shutdown Program   ||\n";
        cout << "|| 3. Logout             ||\n";
        cout << "==========================\n";
        cout << "Select: ";
        cin >> choice;

        //Allows admin to change stock quantities
        if (choice == 1) {
            for (int i = 0; i < 3; i++)
                cout << "(" << i + 1 << ") " << product[i] << " - Stocks: " << stocks[i] << endl;

            int item, newStock;
            cout << "Select which item (1-3): ";
            cin >> item;

            if (item >= 1 && item <= 3) {
                cout << "Enter new stock: ";
                cin >> newStock;
                if (newStock >= 0) stocks[item - 1] = newStock;
            } else {
                cout << "Invalid item.\n";
            }

        //Shutdowns system
        } else if (choice == 2) {
            programRunning = false;
            cout << "\nSystem will shutdown...\n";
        }
    } while (choice != 3 && programRunning); //Keep menu running until logout
}
```

Figure 13. Manager Menu

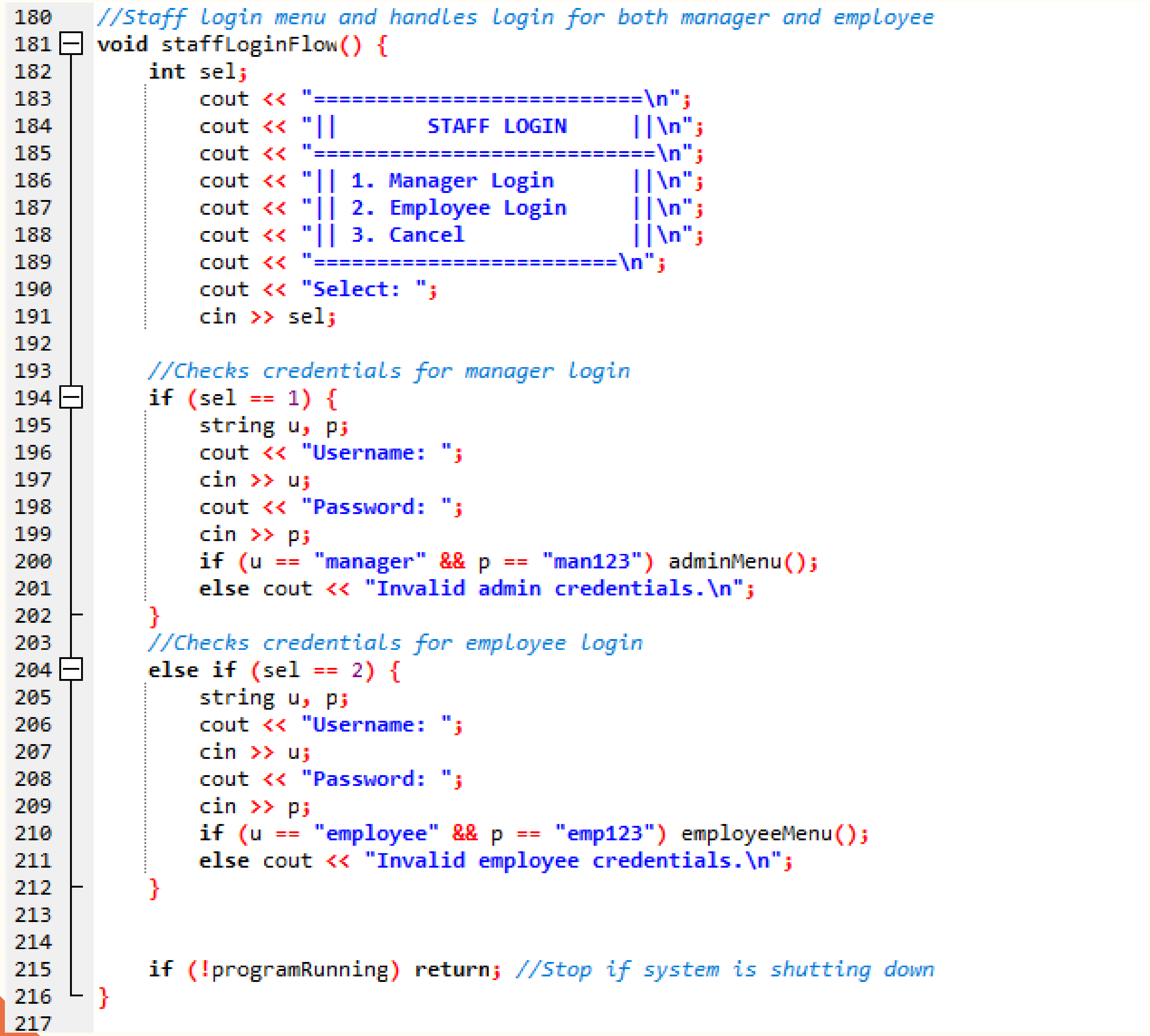
CODE



```
//Verifies manager login to allow discount
bool verifyDiscountAccess() {
    string username, password;
    while (true) {
        cout << "\nManager/Admin Login Required\n";
        cout << "Username: ";
        cin >> username;
        cout << "Password: ";
        cin >> password;

        //hardcoded credentials
        if  (username == "admin" && password == "admin123")
            return true;
        else if (username == "manager" && password == "man123")
            return true;

        char retry;
        cout << "Invalid login. Try again? (y/n): ";
        cin >> retry;
        if (retry == 'n' || retry == 'N') return false;
    }
}
```

Figure 14. Discount Access

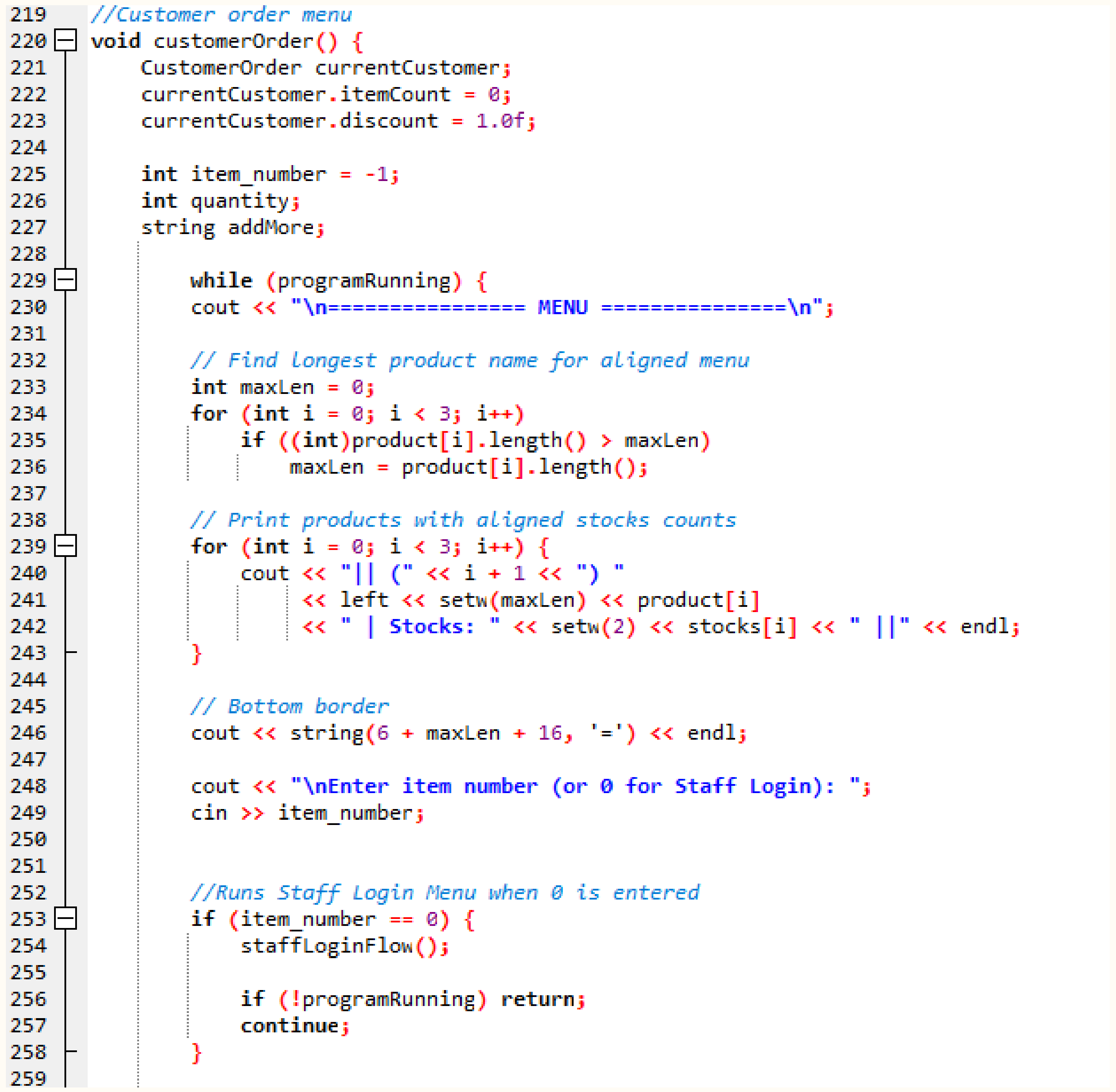
CODE



```cpp
//Staff login menu and handles login for both manager and employee
void staffLoginFlow() {
    int sel;
        cout << "==========================\n";
        cout << "||       STAFF LOGIN     ||\n";
        cout << "===========================\n";
        cout << "|| 1. Manager Login      ||\n";
        cout << "|| 2. Employee Login     ||\n";
        cout << "|| 3. Cancel             ||\n";
        cout << "========================\n";
        cout << "Select: ";
        cin >> sel;

    //Checks credentials for manager login
    if (sel == 1) {
        string u, p;
        cout << "Username: ";
        cin >> u;
        cout << "Password: ";
        cin >> p;
        if (u == "manager" && p == "man123") adminMenu();
        else cout << "Invalid admin credentials.\n";
    }
    //Checks credentials for employee login
    else if (sel == 2) {
        string u, p;
        cout << "Username: ";
        cin >> u;
        cout << "Password: ";
        cin >> p;
        if (u == "employee" && p == "emp123") employeeMenu();
        else cout << "Invalid employee credentials.\n";
    }


    if (!programRunning) return; //Stop if system is shutting down
}
```

Figure 15. Staff Login Menu

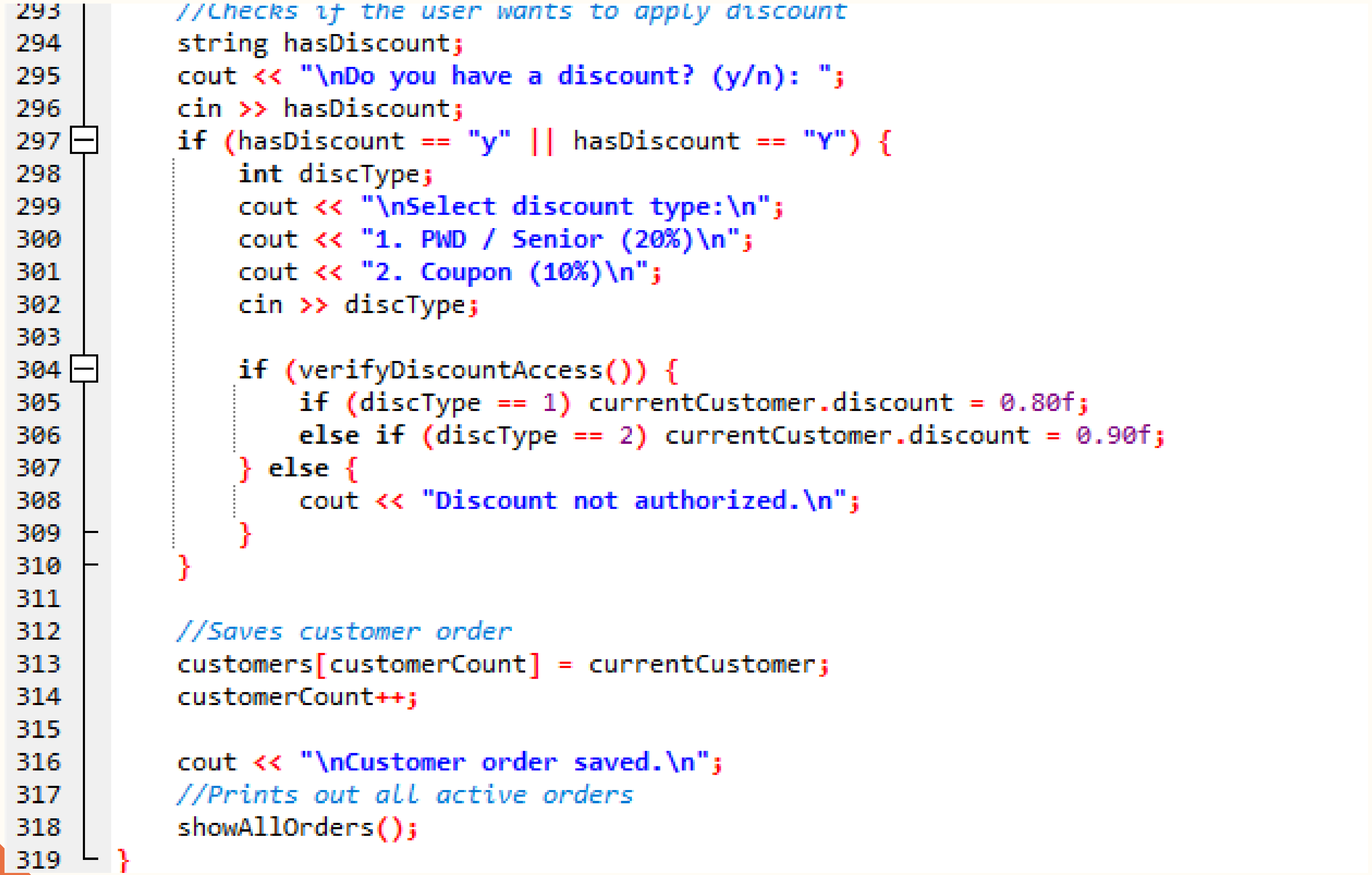
CODE



```
//Customer order menu
void customerOrder() {
    CustomerOrder currentCustomer;
    currentCustomer.itemCount = 0;
    currentCustomer.discount = 1.0f;

    int item_number = -1;
    int quantity;
    string addMore;

        while (programRunning) {
        cout << "\n=============== MENU ==============\n";

        // Find longest product name for aligned menu
        int maxLen = 0;
        for (int i = 0; i < 3; i++)
            if ((int)product[i].length() > maxLen)
                maxLen = product[i].length();

        // Print products with aligned stocks counts
        for (int i = 0; i < 3; i++) {
            cout << "|| (" << i + 1 << ") "
                 << left << setw(maxLen) << product[i]
                 << " | Stocks: " << setw(2) << stocks[i] << " ||" << endl;
        }

        // Bottom border
        cout << string(6 + maxLen + 16, '=') << endl;

        cout << "\nEnter item number (or 0 for Staff Login): ";
        cin >> item_number;


        //Runs Staff Login Menu when 0 is entered
        if (item_number == 0) {
            staffLoginFlow();

            if (!programRunning) return;
            continue;
        }

```

Figure 16. Customer Order Menu

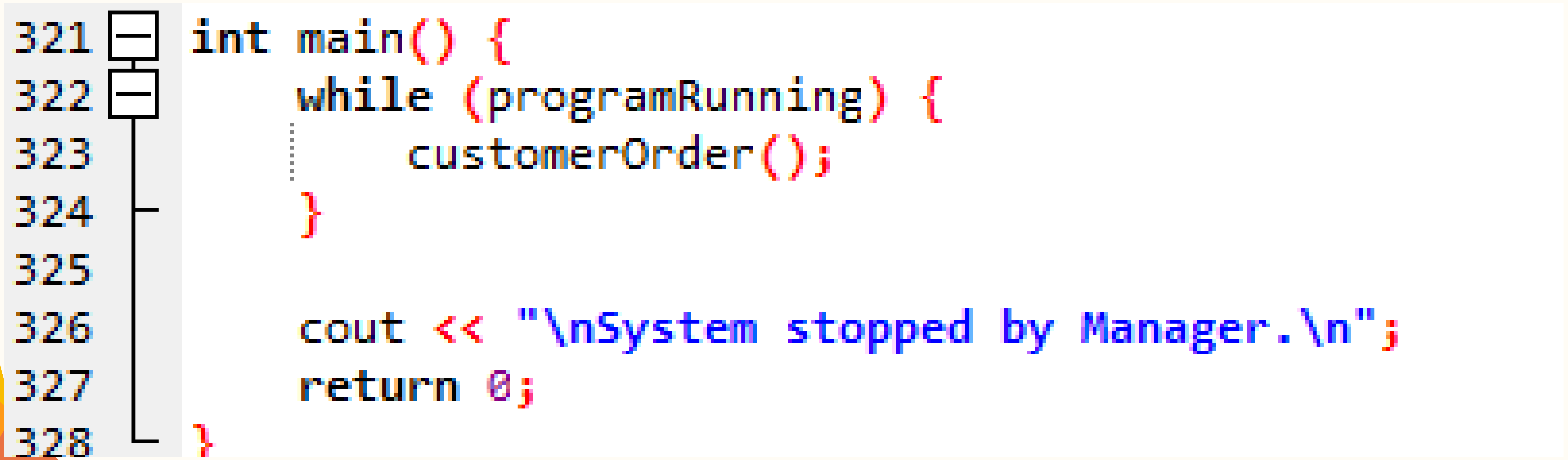
CODE



```
    //Checks if the user wants to apply discount
    string hasDiscount;
    cout << "\nDo you have a discount? (y/n): ";
    cin >> hasDiscount;
    if (hasDiscount == "y" || hasDiscount == "Y") {
        int discType;
        cout << "\nSelect discount type:\n";
        cout << "1. PWD / Senior (20%)\n";
        cout << "2. Coupon (10%)\n";
        cin >> discType;

        if (verifyDiscountAccess()) {
            if (discType == 1) currentCustomer.discount = 0.80f;
            else if (discType == 2) currentCustomer.discount = 0.90f;
        } else {
            cout << "Discount not authorized.\n";
        }
    }

    //Saves customer order
    customers[customerCount] = currentCustomer;
    customerCount++;

    cout << "\nCustomer order saved.\n";
    //Prints out all active orders
    showAllOrders();
}
```

Figure 17. Discount Checker

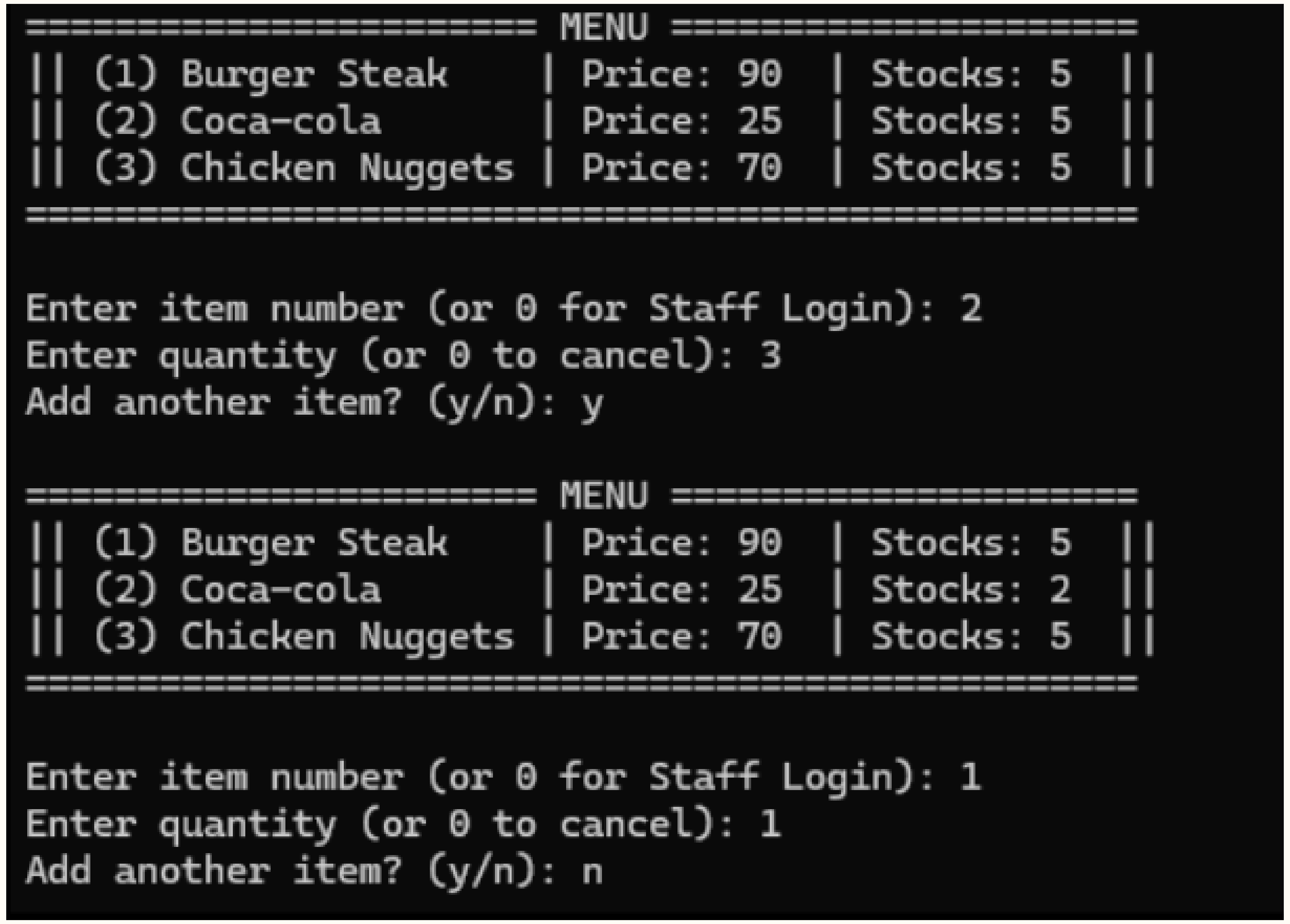
CODE



```cpp
int main() {
    while (programRunning) {
        customerOrder();
    }

    cout << "\nSystem stopped by Manager.\n";
    return 0;
}
```

Figure 18. Program Flow

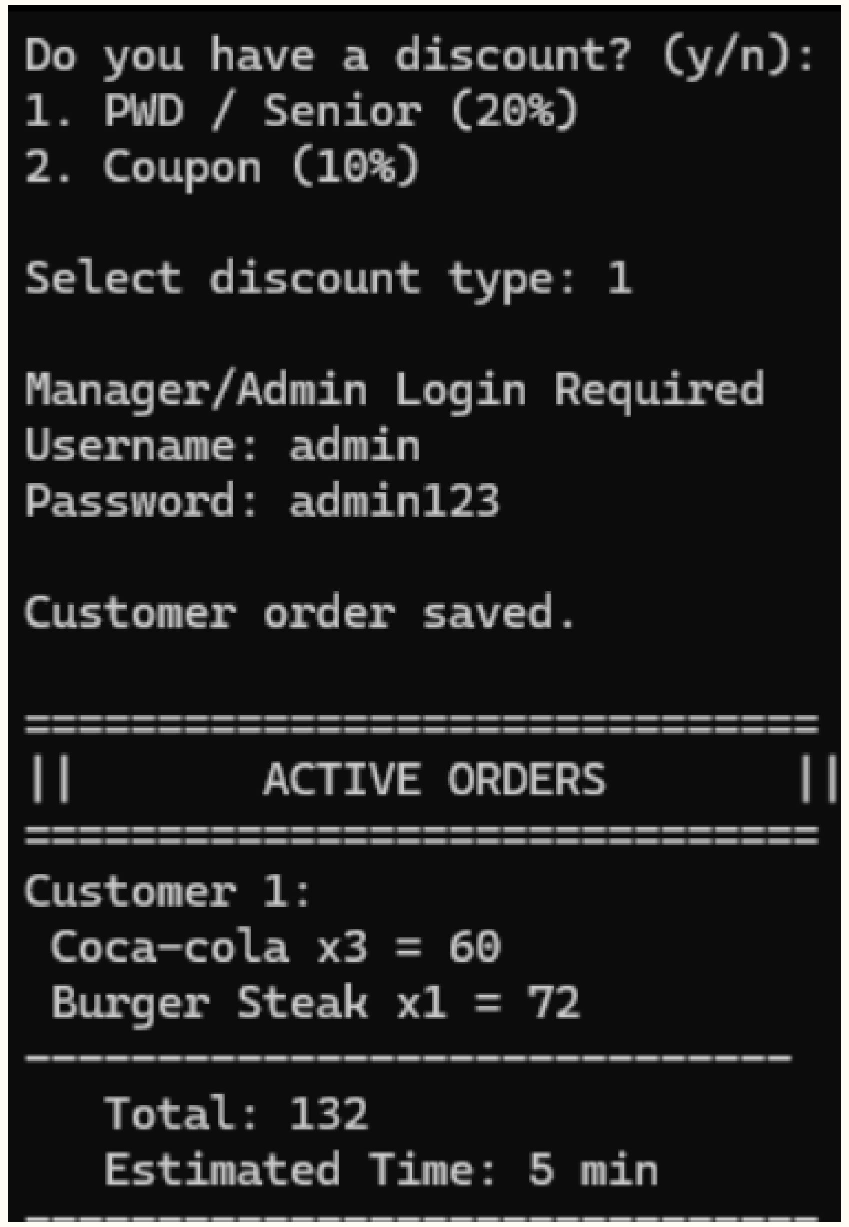
RESULTS AND DISCUSSION



```
======================= MENU =====================
|| (1) Burger Steak     | Price: 90  | Stocks: 5  ||
|| (2) Coca-cola        | Price: 25  | Stocks: 5  ||
|| (3) Chicken Nuggets  | Price: 70  | Stocks: 5  ||
==================================================

Enter item number (or 0 for Staff Login): 2
Enter quantity (or 0 to cancel): 3
Add another item? (y/n): y

======================= MENU =====================
|| (1) Burger Steak     | Price: 90  | Stocks: 5  ||
|| (2) Coca-cola        | Price: 25  | Stocks: 2  ||
|| (3) Chicken Nuggets  | Price: 70  | Stocks: 5  ||
==================================================

Enter item number (or 0 for Staff Login): 1
Enter quantity (or 0 to cancel): 1
Add another item? (y/n): n
```

This output presents the customer menu interface, demonstrating how customers select items, view prices and stock, and input their designed quantities

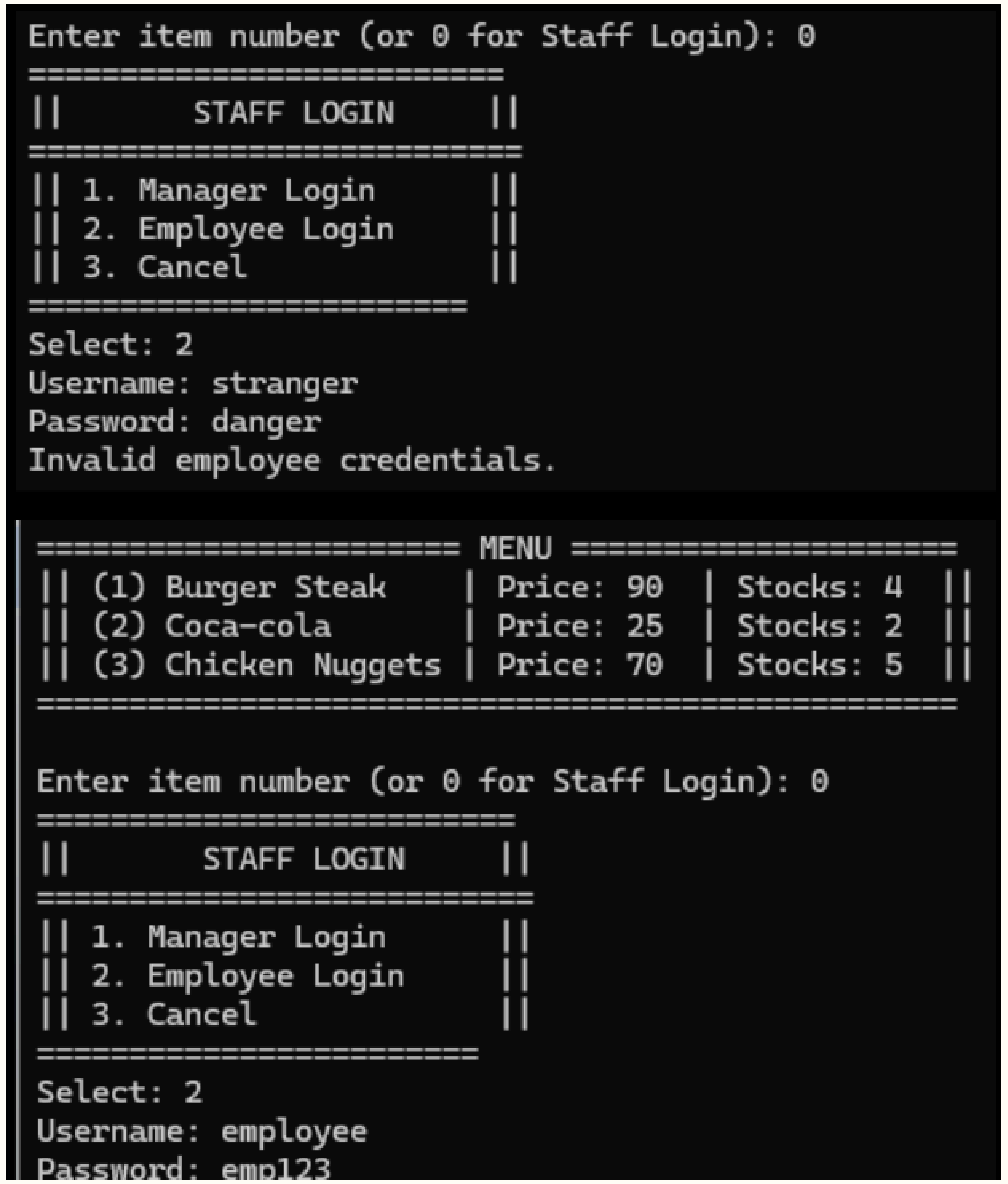
RESULTS AND DISCUSSION



```
Do you have a discount? (y/n):
1. PWD / Senior (20%)
2. Coupon (10%)

Select discount type: 1

Manager/Admin Login Required
Username: admin
Password: admin123

Customer order saved.

==============================
||        ACTIVE ORDERS       ||
==============================
Customer 1:
 Coca-cola x3 = 60
 Burger Steak x1 = 72
-----------------------------
   Total: 132
   Estimated Time: 5 min
```

The output focuses on the discount part of the system, where a manager verifies the discount to ensure it is valid and properly applied to the customer's order. It also prints out the order summary with the items selected, quantity, price with or without discount, total and estimated time for preparation.

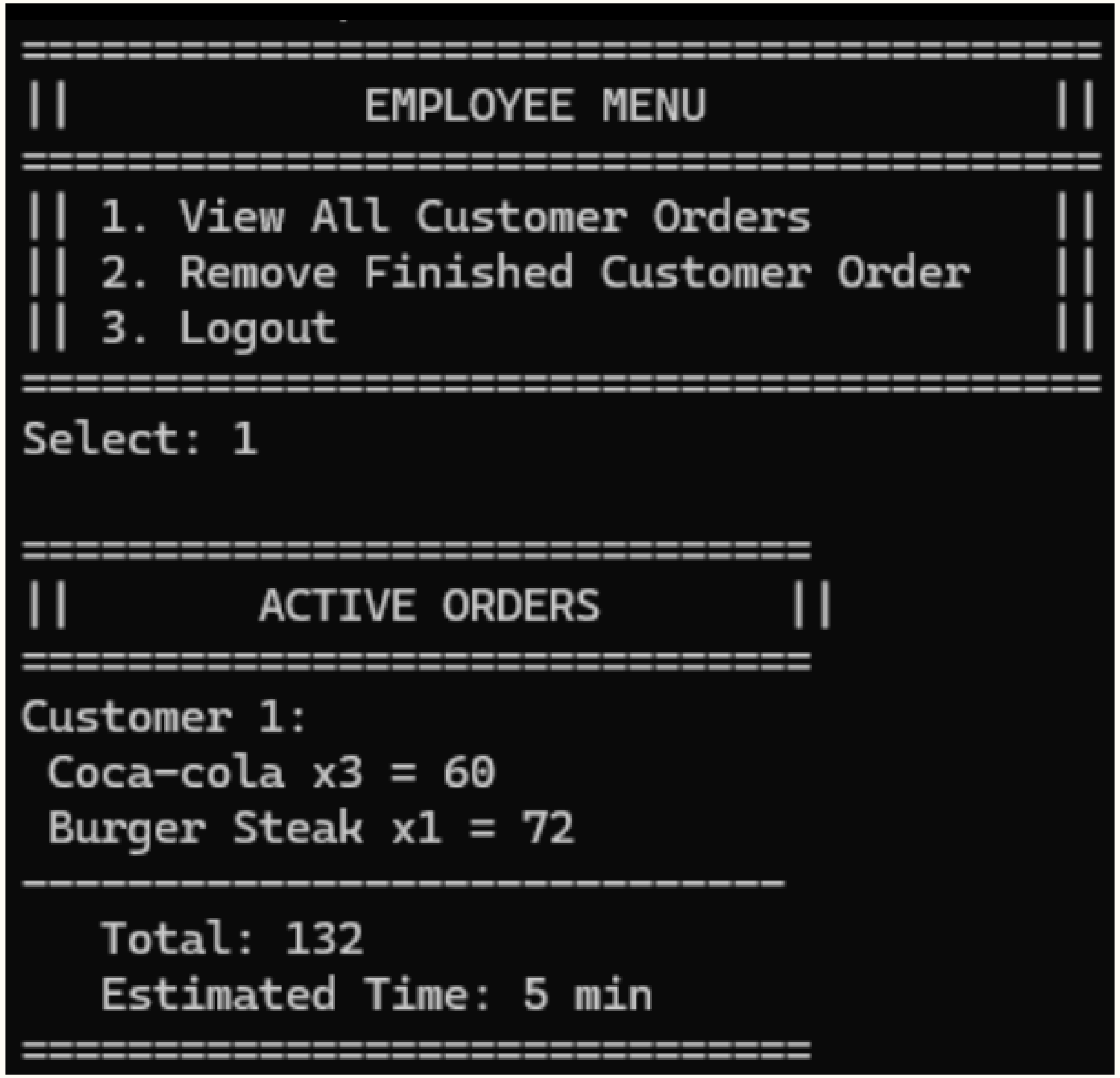
RESULTS AND DISCUSSION



```
Enter item number (or 0 for Staff Login): 0
==========================
||       STAFF LOGIN     ||
===========================
|| 1. Manager Login      ||
|| 2. Employee Login     ||
|| 3. Cancel             ||
========================
Select: 2
Username: stranger
Password: danger
Invalid employee credentials.

======================= MENU =====================
|| (1) Burger Steak     | Price: 90  | Stocks: 4  ||
|| (2) Coca-cola        | Price: 25  | Stocks: 2  ||
|| (3) Chicken Nuggets  | Price: 70  | Stocks: 5  ||
==================================================

Enter item number (or 0 for Staff Login): 0
==========================
||       STAFF LOGIN     ||
===========================
|| 1. Manager Login      ||
|| 2. Employee Login     ||
|| 3. Cancel             ||
========================
Select: 2
Username: employee
Password: emp123
```

This screenshot focuses on the staff login for employees, ensuring that the credentials entered are correct before granting access to the staff menu

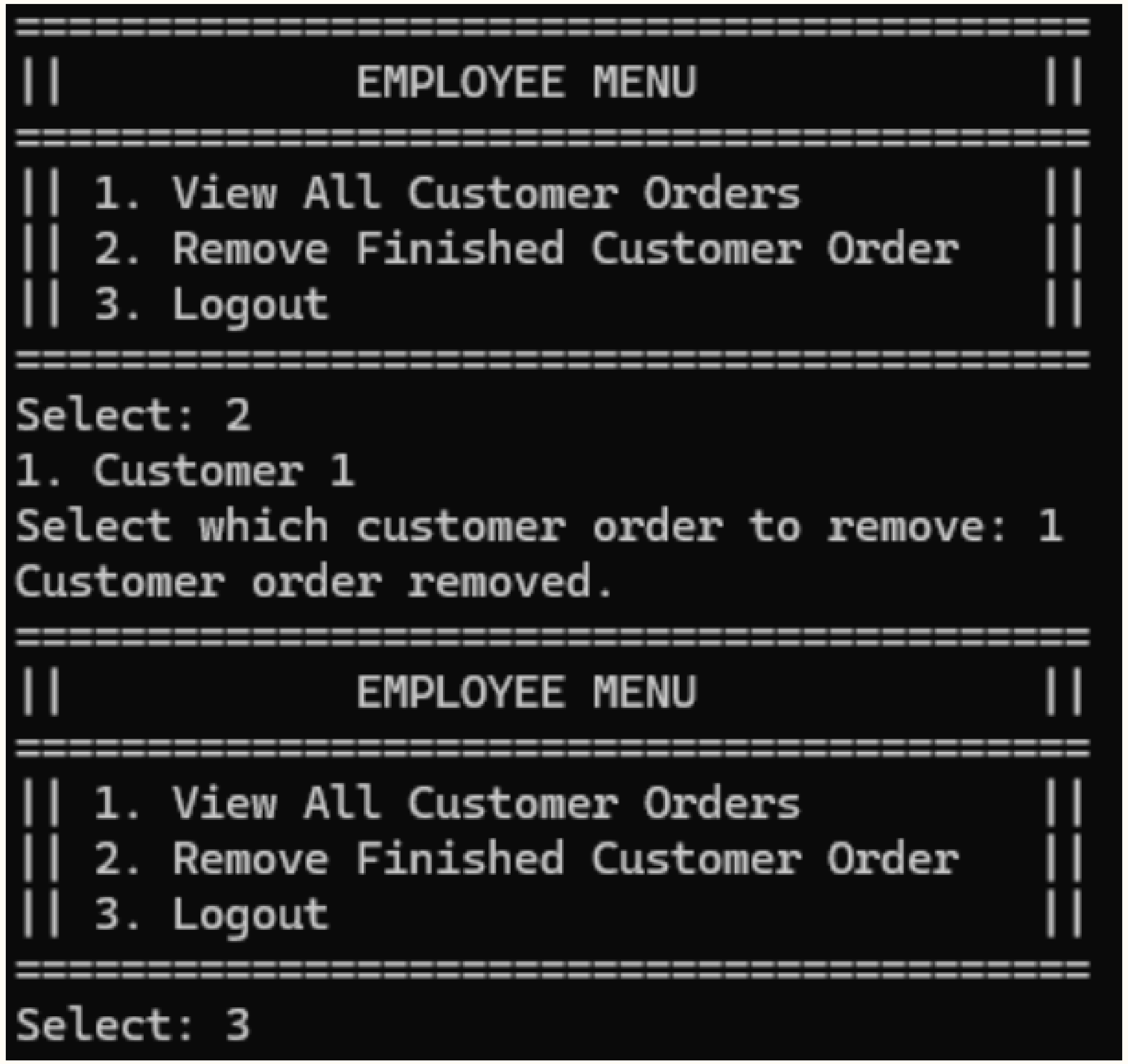
RESULTS AND DISCUSSION



```
==========================================
||             EMPLOYEE MENU            ||
==========================================
|| 1. View All Customer Orders          ||
|| 2. Remove Finished Customer Order    ||
|| 3. Logout                            ||
==========================================
Select: 1

==============================
||       ACTIVE ORDERS      ||
==============================
Customer 1:
 Coca-cola x3 = 60
 Burger Steak x1 = 72
-----------------------------
   Total: 132
   Estimated Time: 5 min
==============================
```

The output demonstrates one of the functions of the employee menu, viewing all active customer orders.

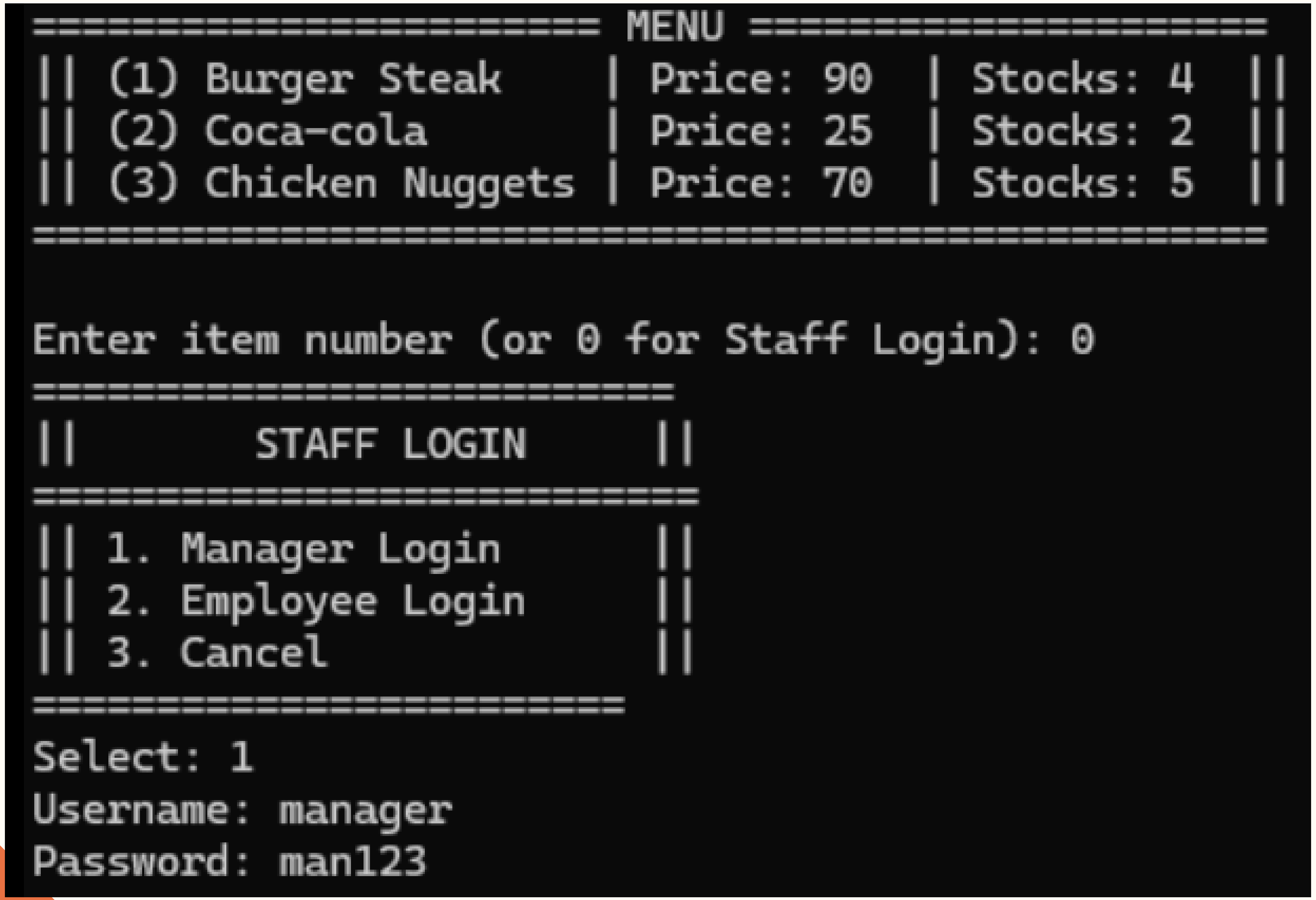
RESULTS AND DISCUSSION



```
=========================================
||            EMPLOYEE MENU            ||
=========================================
|| 1. View All Customer Orders         ||
|| 2. Remove Finished Customer Order   ||
|| 3. Logout                           ||
=========================================
Select: 2
1. Customer 1
Select which customer order to remove: 1
Customer order removed.
=========================================
||            EMPLOYEE MENU            ||
=========================================
|| 1. View All Customer Orders         ||
|| 2. Remove Finished Customer Order   ||
|| 3. Logout                           ||
=========================================
Select: 3
```

The output demonstrates the second function of the employee menu, removing finished customer orders

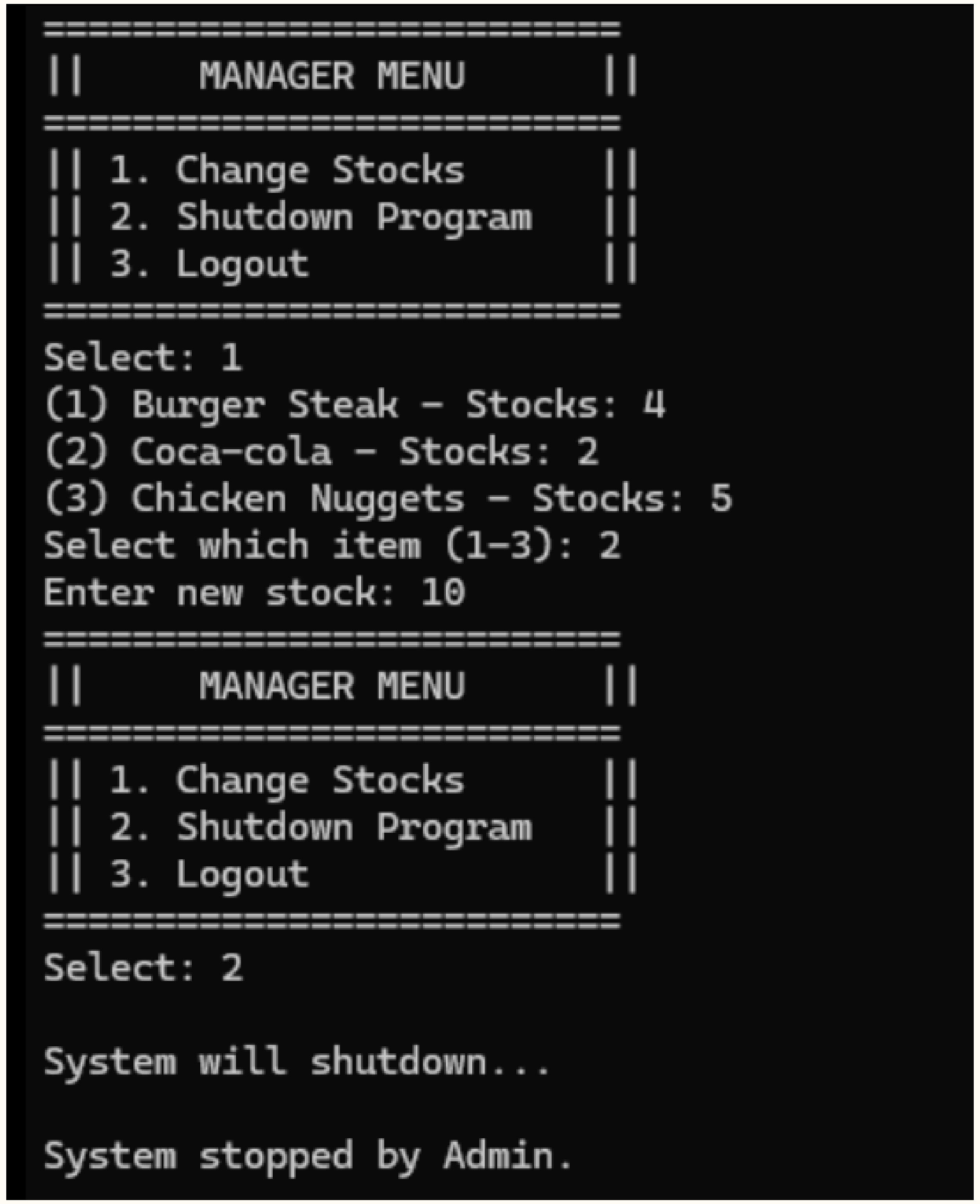
RESULTS AND DISCUSSION



```
======================= MENU =====================
|| (1) Burger Steak      | Price: 90  | Stocks: 4  ||
|| (2) Coca-cola         | Price: 25  | Stocks: 2  ||
|| (3) Chicken Nuggets   | Price: 70  | Stocks: 5  ||
==================================================


Enter item number (or 0 for Staff Login): 0
==========================
||       STAFF LOGIN     ||
===========================
|| 1. Manager Login      ||
|| 2. Employee Login     ||
|| 3. Cancel             ||
========================
Select: 1
Username: manager
Password: man123
```

This output demonstrates the manager login.

RESULTS AND DISCUSSION



```
=========================
||     MANAGER MENU     ||
=========================
|| 1. Change Stocks     ||
|| 2. Shutdown Program  ||
|| 3. Logout            ||
=========================
Select: 1
(1) Burger Steak - Stocks: 4
(2) Coca-cola - Stocks: 2
(3) Chicken Nuggets - Stocks: 5
Select which item (1-3): 2
Enter new stock: 10
=========================
||     MANAGER MENU     ||
=========================
|| 1. Change Stocks     ||
|| 2. Shutdown Program  ||
|| 3. Logout            ||
=========================
Select: 2

System will shutdown...

System stopped by Admin.
```

The output demonstrates the functions of the admin, changing the stocks and shutting the system down.

CONCLUSION

The fast food ordering system demonstrates how programming concepts such as arrays and loops can be applied to address real-world challenges in the food service industry. Based on the results and discussion, the system successfully performed its main functions, including customer ordering, discount validation, staff login, and managerial controls. It also provided accurate calculations of total price, discounts, and estimated waiting time, resulting in faster and more organized transactions.

To further improve the system, it is recommended to enhance the user interface and integrate additional features for practical use. Future updates could include online ordering via mobile or web platforms, real-time sales and inventory tracking, and automated receipt printing to improve customer convenience.

References

Pangilinan, B. (2025, May 28). Digital transformation of the order-taking process in fast food restaurants in the Philippines. Digital Archives @ UP Diliman.

https://digitalarchives.upd.edu.ph/item/62984/1091?fbclid=IwY2xjawOBi05leHRuA2FlbQIxMABicmlkETFwOFdtQWJ5cHJvN0g1ME5Ec3J0YwZhcHBfaWQQMjIyMDM5MTc4ODIwMDg5MgABHp5N7Uny7zI9_rPTm5IQa05roj18UJElHd_xTa23e0Ap9v4sA_uEuI1DEeKv_aem_WK3yjthc4Ueorb5zsFUKsA

https://www.w3schools.com/cpp/cpp_variables.asp
W3Schools.com. https://www.w3schools.com/cpp/default.asp